# Placement simultané en translation

Jean-Daniel Boissonnat, Francis Avnaim

INRIA

UNITÉ DE RECHERCHE
INRIA-SOPHIA ANTIPOLIS

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél.:(1) 39 63 55 11

Rapports de Recherche

N° 689

# PLACEMENT SIMULTANE EN TRANSLATION

Francis AVNAIM
Jean-Daniel BOISSONNAT

JUIN 1987

Mai 1987

# PLACEMENT DE FORMES POLYGONALES: 1.

# PLACEMENT SIMULTANE EN TRANSLATION

**Francis Avnaim et Jean-Daniel Boissonnat**

INRIA
Avenue Emile Hugues
06565 VALBONNE

## RESUME

Nous nous intérressons au placement en translation et sans recouvrement de plusieurs polygones $I_1 ... I_k$ dans un polygone E. Dans le cas où k=1, nous résolvons le problème le plus général, c'est à dire celui où I et E sont des régions polygonales. L'algorithme proposé trouve l'ensemble de tous les placements possibles et sa complexité est proche de l'optimum dans le cas le pire. Nous résolvons également le problème du placement simultané de deux polygones dans un autre et de trois polygones dans un parallèlogramme. Les algorithmes proposés calculent un placement s'il en existe un ou bien calculent une représentation implicite de l'ensemble de tous les placements possibles. Tous les algorithmes utilisent un petit nombre de primitives géométriques qui peuvent être implémentées sans grandes difficultés. Des résultats expérimentaux sont donnés à la fin de l'article. Finalement, nous montrons que les méthodes utilisées dans le plan se généralisent au cas des polyèdres.

May 1987

# THE POLYGON CONTAINMENT PROBLEM : 1.

# SIMULTANEOUS CONTAINMENT UNDER TRANSLATION

**Francis Avnaim and Jean-Daniel Boissonnat**

INRIA
Avenue Emile Hugues
06565 VALBONNE

**ABSTRACT**

We investigate the problem of whether one or several polygons $I_1 ... I_k$ can be translated to fit inside another polygon E without overlapping each other. In case that k=1, we present an algorithm that solves the general case where I and E may be non-convex, may have (non-convex) holes and even may be non-connected, i.e. may consist of several disjoint polygons. The algorithm provides the set of all the possible placements and is close to optimal in the worst-case. We derive also algorithms which simultaneously fit two or three polygons inside another one. In case of two polygons, the polygons may be non-convex. In case of three polygons, E is restricted to be a parallelogram. We present algorithms that only find one feasible placement if one exists, and algorithms that find an implicit representation of the whole feasible region. All the algorithms make use of only a small set of primitives that can be implemented rather easily. Experimented results are discussed. Finally, we show that our methods can be generalized to the case of polyhedra .

## I- INTRODUCTION

We investigate the problem of whether one or several polygons $I_1 ... I_k$ can be translated to fit inside another polygon E without overlapping each other. This problem has been extensively studied in the past for applications involving only a limited class of shape-types, such as rectangles. The earliest investigations considered the case where E, $I_1,...,I_k$ are all rectangles with the same axis. Such a problem appears, for example, in the cutting of steel, wood or glass plates. For small problems, most authors use linear programming and dynamic programming techniques (GG1-3). For larger problems, these techniques do not produce a solution within acceptable time and tree search and heuristics have been proposed (AA,FZ,H,HF). These techniques do not apply to more general shapes. In case that E is a sufficiently large rectangle and that the $I_i$ are irregular shapes without holes, Albano and Sappupo have proposed a heuristic search method (AS). In some situations, however, we have to deal with the case where E and the $I_i$ are irregular shapes, possibly with holes. Moreover, the $I_i$ cannot always be assumed to be much smaller than E, so that approximate and heuristic methods would presumably fail to find a solution. Such a situation appears in the leather industry: the leather sheets tend to be highly irregular in shape and contain defective areas that cannot be used. Only a few pieces (usually four or five) can be cut out of the leather sheet. In order to solve such a problem we need a more geometric approach.

Non-heuristic solutions to the problem for *one* polygon, say I, has been studied previously in the Computational Geometry literature(C, BFM, F). Chazelle derived an algorithm that runs in time O(n+m) to solve this problem, in the case that both polygons E and I are convex. Here n is the number of edges of E and m the number of edges of I. The case that I is non-convex but E is convex, can be easily reduced to the previous one. The case that E is non-convex but I is convex is considered in (BFM) and (F). The best algorithm is obtained by Fortune who presents an O(nm lognm) algorithm. This algorithm is surely close to optimal (in the worst-case sense) since the feasible region may have O(mn) vertices. This paper presents an algorithm that solves the general case where I and E may be non-convex, may have (non-convex) holes and even may be non-connected, i.e. may consist of several disjoint polygons. The algorithm provides the set of all the possible placements. Its complexity is $O(c_i c_e N \log N)$ where $N=c_i n+c_e m$, $c_i$=1+number of concave vertices of I, $c_e$=number of convex vertices of E which are not vertices of the convex hull of E plus number of edges of the convex hull of E which are not edges of E . This algorithm is close to optimal, in the worst-case where $c_e=O(n)$ and $c_i=O(m)$, since the feasible region may have $O(n^2m^2)$ vertices. In the case of rectilinear polygons the algorithm runs in $O(n^2m^2)$ time and, thus, is optimal.

The problem of simultaneously fitting several polygons inside another one has not received much attention in the computational geometry literature. Dori and Ben-Bassat deal with the nesting of congruent replications of a given convex figure within a large rectangular board (DBB). Guibas, Ramshaw and Stolfi give a necessary and sufficient condition to decide in linear time if two convex polygons can be made to be disjointly fit into a third convex one. In this paper, we derive algorithms that compute a feasible placement for two and three polygons to fit inside E without overlapping each other. In case of two polygons, the polygons may be non-convex, possibly non-connected and with holes. In case of three polygons, we present an algorithm when E is a parallelogram. All the methods we present in this paper can be generalized easily to the case of polyhedra though the complexity of the algorithms and the difficulty to code them up increase significantly.

A related problem to the polygon containment problem is to find a collision free path for a polygonal object. This problem has been extensively studied since the early work of Lozano-Pérez and Wesley (LPW). A survey can be found in (W) and a collection of recent results in (S). The problem of coordinating the movements of several disks has been studied by Schwartz and Sharir (SS) and by Yap(Y). Yap has obtained an $O(n^k)$ algorithm for the coordination of k disks (k=2 or 3) inside a polygon with n edges. The algorithm computes only a subset of the feasible region and then uses a graph search algorithm to find the desired motions. The construction of the graph is closely related to our problem. The graph represents an appropriate subset of the free placements, sufficient to guarantee the finding of a path if one exists; in the polygon containment problem, we are interested in two extreme instances of the same basic problem: we may be interested in finding the whole set of feasible placements or, on the other hand, we may be interested in simply reporting one feasible placement if one exists. Both problems are considered in this paper.

## 2- THE CONTAINMENT PROBLEM FOR ONE POLYGON

### 2-1 Notations

All the polygons are supposed to be defined in a reference frame. In the sequel, we identify a point M in the plane and the corresponding vector **OM** where O is the origine of the reference frame. Thus a polygon will be considered either as a set of points or as a set of vectors. We note C(P,Q) the set of translations that fit P inside Q and O(P,Q) the set of translations that make P to intersect Q. Note that C(P,Q) and O(P,Q) are related by $C(P,Q)=\overline{O(P,\bar{Q})}$ if we note $\bar{A}$ the set of points which do not belong to A. We note CH(P) the convex hull of polygon P, and $T_u$ the translation of vector u. If A and B are two sets, A - B denotes the set of elements which belong to A but not to B .

In the sequel, we want to fit polygon I inside polygon E. I has m edges and we note $c_i$ the number of concave vertices of I plus one. E has n edges. If E is non-convex, CH(E) - E consists of a number, say $p_e$, of disjoint polygons, the *pockets* of E. Let $v_e$ be the number of concave vertices of theses pockets (equivalently, $v_e$ is the number of convex vertices of E which are not vertices of the convex hull of E). We note $c_e=v_e+p_e$.

### 2-2 Generalities

Our approach follows Baker, Fortune and Mahaney one (BFM). We reduce the general problem to several subproblems, where only convex polygons are manipulated; then we combine the partial results in order to obtain the solution. We recall now some results which will be useful in the sequel.

**Proposition 1** (C): *if E and I are convex, C(I,E) is a convex polygon with, at most, n edges that can be computed in O(n+m) time.*

**Proposition 2** (C): *if E is convex, C(I,E)=C(CH(I),E).*

**Proposition 3** (GRS): *if E and I are convex, O(I,E) is a convex polygon with, at most, n+m edges that can be computed in O(n+m) time.*

**Proposition 4 (PS, OWW, WW):** *let P and Q be two polygons with p and q edges respectively. Let N=p+q. If P and Q are convex, Boolean operations on P and Q can be done in time O(N). If P or Q is non-convex, Boolean operations can be done in time O(NlogN). If both P and Q are non-convex, Boolean operations can be done in time O((N+t)logN) where t is the number of intersections between the edges of P and Q. In case that P and Q are rectilinear, the complexity is reduced to O(NlogN+s), which is optimal. Here s is the number of edges of the output.*

**Corollary:** *Let $P_1...P_k$ be a bounded number k of polygons whose total number of edges is N. If the $P_i$ are convex, Boolean operations on the $P_i$ can be done in O(NlogN) time. Otherwise, Boolean operations can be done in O((N+t)logN) time if t is the number of intersections between the edges of the $P_i$.*

**Proposition 5 (KS):** *Let T be a set of k convex polygons whose interior do not overlap and let t be the total number of edges of T. If I is a convex polygon with m edges, O(I,T) consists of a set of polygons with O(km+t) edges. It can be computed in O((km+t)log(km+t)logk).*

**Proposition 6 (HM):** *A simple polygon P, with p edges and c concave vertices, can be decomposed into fewer than 2c+1 convex subpolygons in time O(p logp).*

### 2-3 Containment of a convex polygon

In this section E is non convex and I is convex ($c_i$=1). E will be considered as a convex polygon (CH(E)) containing a set of disjoint polygonal holes, the pockets of E (CH(E) - E). The following result holds:

**Theorem 1:** *The set S of all valid placements is given by: S=C(I,CH(E)) - O(I,E - CH(E)).*

In order to reduce the problem to convex subproblems only, we decompose the pockets of E (CH(E) - E) into convex parts, say $T_1...T_k$. These $T_i$ are considered as holes inside CH(E). Thus $S = C(I,CH(E)) - O(I, \cup_i T_i) = C(I,CH(E)) - \cup_i O(I,T_i)$. The algorithm is described below:

**Algorithm NCC**

1- Compute CH(E);

2- Decompose CH(E) - E into convex parts; /*noted $T_1 ... T_k$*/

3- Compute W=C(I,CH(E));

4- Compute $O=\cup_i O(I,T_i)$;

5- **Return** S=W - O;

6- **End.**

Step 1 takes O(n) time (PS). CH(E) - E is a set of $p_e$ disjoint polygons with, at most, n edges and $v_e$ concave vertices. Once step 1 has been performed, it can be decomposed into $k \leq 2c_e - p_e < 2c_e$ convex parts in O(n logn) time (Proposition 6). Step 3 takes O(n+m) time (Proposition 1). Due to Proposition 5, step 4 takes O((km+n)log(km+n)logk) which is less than $O((c_e m+n)\log(c_e m+n)\log c_e)$. O is a set of disjoint polygons with, at most, $O(c_e m+n)$ edges; W is a convex polygon with, at most, n edges. So, due to Proposition 4, step 5 can be done in $O((c_e m+n)\log(c_e m+n))$. We summarize our results in the following

**Theorem 2**: *the complexity of algorithm NCC is* $O((c_e m+n)\log(c_e m+n)\log c_e)$.

**Remarks:**

1- This algorithm is very similar to Baker, Fortune and Mahaney one. The improvement of the time bound is mainly due to Proposition 5. However, when $c_e=O(n)$, our algorithm is still slower than Fortune's one. Note that $c_e$ can be small compared to n even if E is a very non convex polygon (see fig. 1); on the other hand, it may be of size O(n) for some almost convex polygons (see fig. 2) .

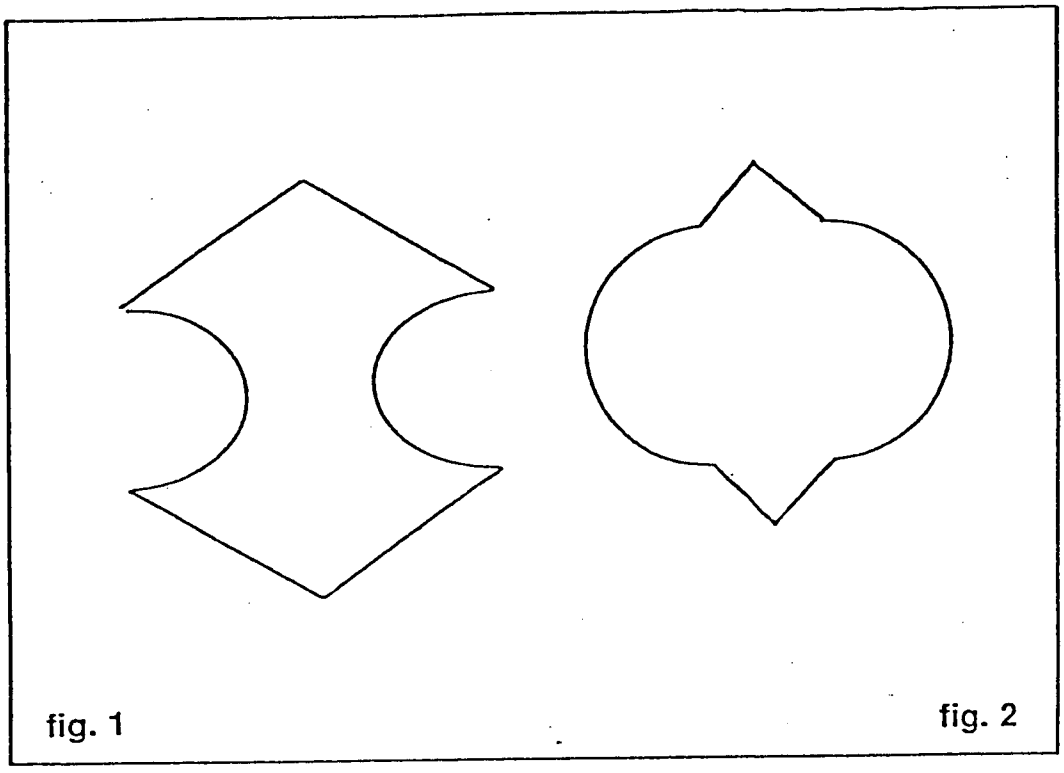

fig. 1 fig. 2

**2-** The key step is the step that computes boolean operations of polygons. We use the algorithm of Ottman, Widmayer and Wood (OWW) which runs in O((N+s)logN) time where N is the total number of edges of the polygons and s the total number of intersections between edges of the polygons. It is not known if this algorithm is optimal. In particular, the question is open whether or not the bound O(NlogN + t) can be reached; t is the number of edges of the output. For the time being, the answer is positive only in the special case of rectilinear polygons (WW). In that case, our algorithm runs in $O((c_e m+n)\log(c_e m+n))$ time and, thus, is as fast as Fortune's one.

**3-** Steps 4 and 5 have been separated for clearness. However the algorithm described in (OWW) can compute directly $W - \cup_i O(I,T_i)$;

**4-** Our algorithm, as the other ones, find all the feasible placements. The question remains open if we can design a less complex algorithm in case we simply want to decide containment. Notice however that we can return a valid placement as soon as an edge of $W- \cup_i O(I,T_i)$ is found. While the worst-case complexity remains the same as for the algorithm which computes all the feasible placements, the average complexity will presumably be greatly reduced.

### 2-4 General case

Our approach is the same as the previous one. The pockets of E are decomposed into convex parts $T_1 ... T_k$. In addition, we decompose also I into convex parts, say $I_1 ... I_l$. A valid placement of C(I,E) is a translation which fits I inside E, thus CH(I) inside CH(E), while preventing one of the $I_i$ to intersect one of the $T_j$:

**Theorem 3**: *The set S of all valid placements is given by:* $S=C(CH(I),CH(E)) - \cup_{ij} O(I_i,T_j)$.

The algorithm is quite similar to the previous one and is roughly described below:

**Algorithm NCNC**

1- Compute CH(E) and decompose CH(E)- E into convex parts; /*we note them $T_1 ... T_k$ */
2- Compute CH(I) and decompose I into convex parts; /*we note them $I_1 ... I_l$ */
3- Compute W=C(CH(I),CH(E)); /*due to Proposition 2, W is equal to C(I,CH(E)) */
4- **For** i=1 to k and **for** j=1 to l compute $O_{ij}=O(I_i,T_j)$;
5- Compute $O=\cup_{ij} O_{ij}$;
6- **Return** W - O;
7- **End.**

Due to Proposition 6, step 1 can be done in O(n logn) time yielding $k \leq 2c_e - p_e < 2c_e$ convex parts. Similarly, step 2 can be done in O(m logm) time yielding $l \leq 2c_i - 1$ convex parts. Due to Proposition 1, step 3 can be done in O(n+m) time. Computing $O_{ij}$ takes $O(n_i+m_j)$, if $n_i$ is the number of edges of $T_i$ and $m_j$ the number of edges of $I_j$. Thus step 4 takes $O(\Sigma_{ij}(n_i+m_j)) = O(c_i n + c_e m)$. Due to Proposition 4, computing O takes $O((c_i n+c_e m+s)\log(c_i n+c_e m))$, if s is the number of intersections between the edges of the $O_{ij}$. Let us evaluate s. $O_{ij}$ is convex and has, at most, $n_i+m_j$ edges. So the number of intersections between $O_{ij}$ and $O_{i'j'}$ is less than $n_i+m_j+n_{i'}+m_{j'}$; thus s is less than $\Sigma_{ii'jj'}(n_i+m_j+n_{i'}+m_{j'})$ which is $O(c_e c_i(c_i n+c_e m))$. We conclude that step 5 requires, at most, $O(c_e c_i(c_i n+c_e m)\log(c_i n+c_e m))$ time. O is a set of disjoint polygons whose total number of edges is at most $O(c_i n+c_e m)$ and W is a convex polygon with at most n edges. Due to Proposition 4, step 6 requires $O((c_i n+c_e m)\log(c_i n+c_e m))$ time. We summarize our results in the following

**Theorem 4**: *The time complexity of the NCNC algorithm is* $O(c_e c_i(c_i n+c_e m)\log(c_i n+c_e m))$.

**Remarks:**

1- In the worst-case where $c_e=O(n)$ and $c_i=O(m)$, the time complexity becomes $O(n^2m^2 \log nm)$ which is close to optimal since the feasible region may have $\Omega(n^2m^2)$ vertices (BFM see fig. 3). As in section 2-3 we notice that $c_e$ may be small compared to n for a class of very non convex polygons .

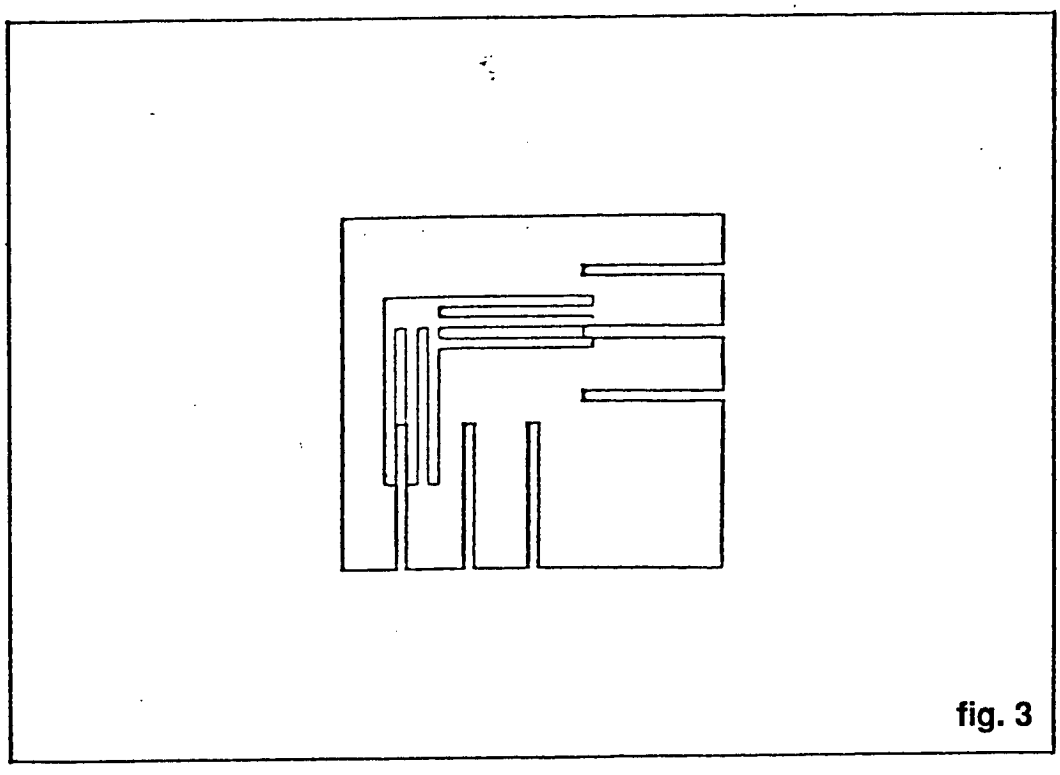

fig. 3

2- Remarks similar to remarks 2 and 3 of the previous section hold. In the case of rectilinear polygons, the complexity of the above algorithm becomes $O(n^2m^2)$, which is worst-case optimal. However the question remains open whether the complexity can be reduced in case we only want to decide containment.

3- The algorithm can be easily extended to the case that E has holes and I must fit inside E but outside the holes. Theorem 4 still holds, provided that n represents the total number of edges of E, including the edges of the holes, and that $c_e$ is incremented by the number of concave vertices of the holes plus the number of holes (as usual, the holes are decomposed into convex parts). The algorithm can even be extended, without increasing the complexity, to the case that E and I consist of several disjoint (but fixed) polygons.

4- Notice that step 5 of algorithm NCNC computes O(I,T) where $T = \cup_j T_j$ . So, the convex decomposition paradigm yields an algorithm to compute O(A,B) in the general case (A and B non convex) . Its complexity is the same as the algorithm NCNC one because step 5 is its most costly step .

## 3-SIMULTANEOUS CONTAINMENT OF SEVERAL POLYGONS : POSITION OF THE PROBLEM

Let E be a polygon and $I_1, .., I_k$ a family of k polygons. The containment problem for the family $I_1, .., I_k$ and E can be enonced as: find a family $T_{p1}, .., T_{pk}$ of translations such that:

$$\forall\ j,\quad E \supset T_{pj}(I_j) \quad \text{and} \quad \forall\ i \neq j,\quad T_{pi}(I_i) \cap T_{pj}(I_j) = \emptyset .$$ (see fig. 4)

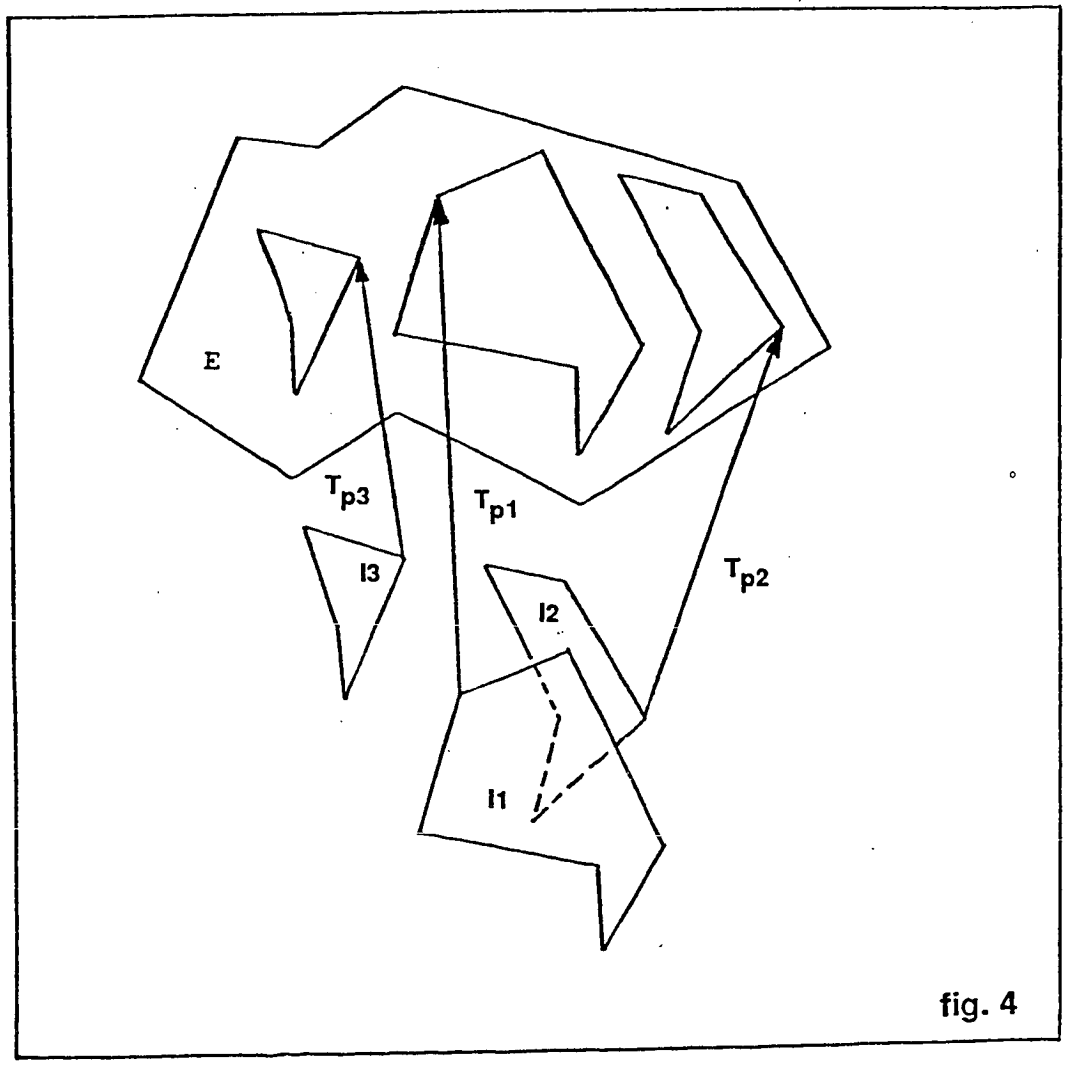


fig. 4

We can see that the problem is of dimension 2k. However, we can decompose the problem into two subproblems of lower dimensions. To do that, we first make some observations.

Let us note $S_i = C(I_i,E)$. It is clear that, when $I_i$ moves, $S_i$ moves too. More precisely, if $I_i$ is translated by vector u, $S_i$ is translated by vector -u, as proved in the following lemma (see fig. 5):

**Lemma 1:** $C(T_u(I_i),E)=T_{-u}(C(I_i,E))=T_{-u}(S_i)$, $i=1,2$.

**proof:** p is a vector of $C(T_u(I_i), E)$ iff $T_p \circ T_u(I_i)$ is included in E which is equivalent to $p+u \in S_i$ or $p \in T_{-u}(S_i)$ . ♦

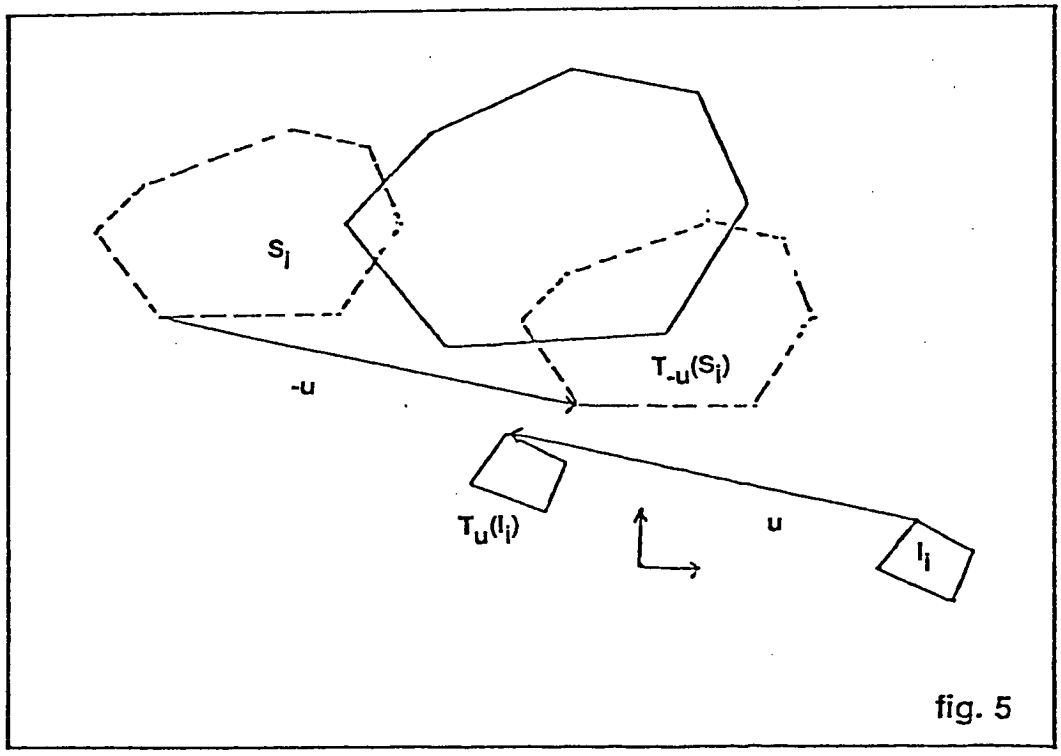


fig. 5

To solve the simultaneous containment problem, we need to translate polygons $I_1$, .., $I_k$, say by vectors $u_1,...,u_k$, such that $T_{u1}(I_1)$, .., $T_{uk}(I_k)$ do not overlap and that there exists a translation $T_p$ which fits $T_{u1}(I_1) \cup T_{u2}(I_2) .. \cup T_{uk}(I_k)$ inside E.

Notice that the previous observations remain true even if one polygon, say $I_1$,is kept fixed ($u_1$=0). Thus the problem can be decomposed in two steps (see fig. 6):

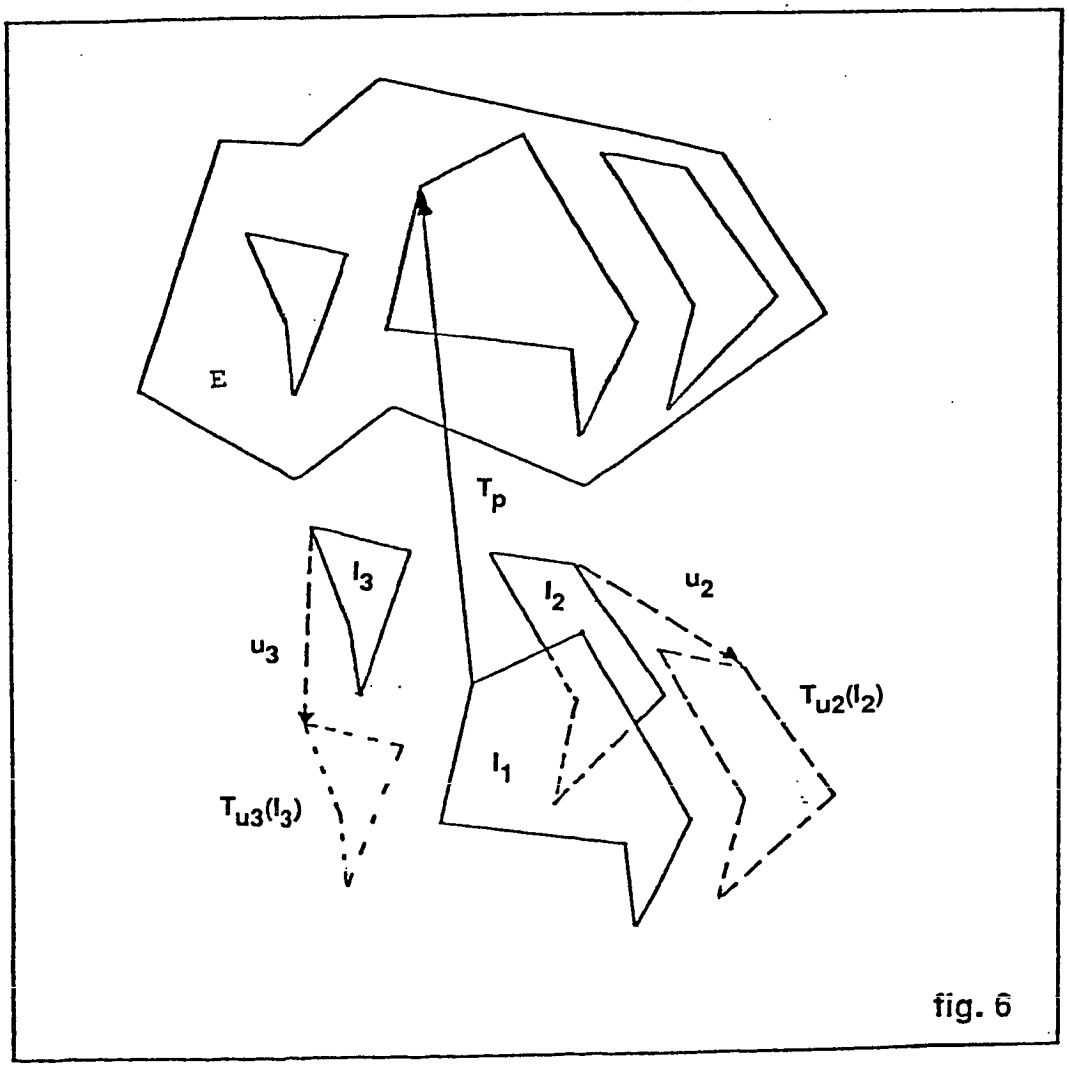


fig. 6

step 1: find all the *relative positions*, i.e. the (k-1)-uples $(u_2,...,u_k)$, such that $I_1$, $T_{u2}(I_2)$, .., $T_{uk}(I_k)$ do not overlap and that there exists a translation $T_p$ which fits $I_1 \cup T_{u2}(I_2) .. \cup T_{uk}(I_k)$ inside E. Such a relative position is said to be a *valid relative position.*

step 2: then for a given valid relative position, find all the valid placements $T_p$.

Step 1 is a 2(k-1) dimensional problem and step 2 is a 2-dimensional one.
Step 2 consists in computing the intersection of several polygons. Indeed, the following lemma holds:

**Lemma 2:** $(u_2, .., u_k)$ is a valid relative position iff $I_1$, $T_{u2}(I_2)$, .., $T_{uk}(I_k)$ do not overlap and $S_1$, $T_{-u2}(S_2)$, .., $T_{-uk}(S_k)$ have a common intersection. Moreover, a translation $T_p$ fits $I_1 \cup T_{u2}(I_2) .. \cup T_{uk}(I_k)$ inside E iff $p \in S_1 \cap T_{-u2}(S_2) .. \cap T_{-uk}(S_k)$.

**proof:** We prove that a translation $T_p$ fits $I_1 \cup T_{u2}(I_2) .. \cup T_{uk}(I_k)$ inside E iff $p \in S_1 \cap T_{-u2}(S_2) .. \cap T_{-uk}(S_k)$. Indeed, any translation of vector p in $S_1 \cap T_{-u2}(S_2) .. \cap T_{-uk}(S_k)$ fits $I_1$, $T_{u2}(I_2)$, .., $T_{uk}(I_k)$ inside E. Conversely if $T_p$ is a translation which fits $I_1 \cup T_{u2}(I_2) .. \cup T_{uk}(I_k)$ inside E, $T_p$ fits $I_1$, $T_{u2}(I_2)$, .., $T_{uk}(I_k)$ inside E thus p is a vector of $S_1$, $T_{-u2}(S_2)$, .., $T_{-uk}(S_k)$ (lemma 1). ◆

Step 1 is more difficult and is solved, in this paper, for the case of two polygons inside another one and for three polygons inside a parallelogram. According to this sheme, the 4-dimensional problem of simultaneously fitting two polygons inside a third one is decomposed in two 2-dimensional problems. Notice that the solution is given in an implicit way: it is not described as a 4-dimensional object, but as 2-dimensional cuts of the 4-dimensional object. The 6-dimensional problem of simultaneously fitting three polygons inside a third one is traited in a similar way. Moreover, we will see how step 1 can be decomposed in two 2-dimensional sub-problems when E is a parallelogram.

## 4- SIMULTANEOUS CONTAINMENT OF TWO POLYGONS

Let E, $I_1$ and $I_2$ be three polygons with n, $m_1$ and $m_2$ edges respectively. We want to fit $I_1$ and $I_2$ inside E in such a way that $I_1$ and $I_2$ do not overlap. A solution to that problem is called a simultaneous placement of $I_1$ and $I_2$ inside E or a placement for short. We assume that neither $S_1=C(I_1,E)$ nor $S_2=C(I_2,E)$ are empty: otherwise no simultaneous placement would be possible.

Guibas, Ramshaw and Stolfi (GRS) have given a necessary and sufficient condition, which can be used to decide in linear time if two convex polygons can be made to disjointly fit into a third convex one . We show that this condition can be extended to non-convex polygons, and it is possible to describe that in an implicit way all the solutions to the problem, without increasing the time complexity (in case of three convex polygons) .

According to the general sheme described in section 3, the method consists in two steps: at first, we compute the valid relative positions of $I_1$ and $I_2$. Then, for a given valid relative position, we compute the set of all the possible placements.

We note $I=O(I_2,I_1)$, $S=O(S_2,S_1)$ and $S^*$ the polygon which is symetric to S with respect to the origin.

**Theorem 6:** *The set U of all the valid relative positions is given by: $U=S^*- I$.*

**Proof:** (see fig.7) A vector u is a valid relative position iff the two conditions below are satisfied ( see lemma 2 of section 3):

(1) $I_1 \cap T_u(I_2) = \emptyset$

(2) $S_1 \cap T_{-u}(S_2) \neq \emptyset$

Condition (1) is equivalent to $u \notin I$. Condition (2) is equivalent to $-u \in S$ or equivalently, $u \in S^*$. In conclusion, both conditions are satisfied iff $u \in S^* - I$ . ♦

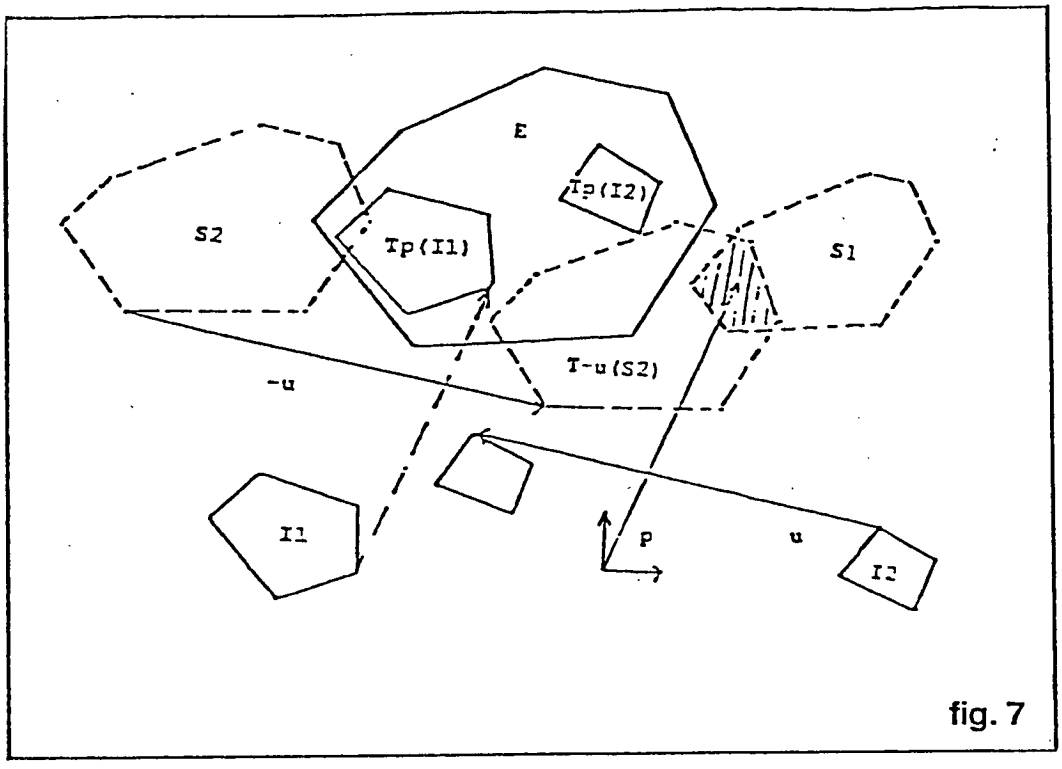


fig. 7

**Remark:**

The edges of U consist of edges which are part of edges of I and $S^*$. The vertices of U are either vertices of I or vertices of $S^*$ or intersections between edges of I and $S^*$. Let M be a point of the boundary of U:

i- if M belongs to an edge of I, M corresponds to a relative position where $I_1$ and $I_2$ are in contact; in the case where M is a vertex of I, this contact is a vertex-vertex contact .

ii- if M belongs to an edge e of $S^*$, M corresponds to a relative position which admits a unique placement where both $I_1$ and $I_2$ are in contact with E (one of them with a double contact with E). In the case where M is a vertex of $S^*$, both $I_1$ and $I_2$ have a double contact with E.

iii- if M is an intersection point between an edge of I and an edge of $S^*$, M corresponds to a relative position which has the two properties above.

An algorithm computing U is the following:

**Algorithm Double containment**

1- Compute $S_1=C(I_1,E)$;

2- Compute $S_2=C(I_2,E)$;

3- **If** $S_1$ or $S_2$ is empty **then** return $U=\emptyset$

**else**

**3-1** Compute $I=O(I_2,I_1)$;

**3-2** Compute $S=O(S_2,S_1)$;

**3-3** Compute $S^*$;

**3-4 Return** $U= S^*- I$.

4- **End.**

This algorithm works for convex or non-convex polygons. If E, $I_1$ and $I_2$ are convex, then the overall complexity of the algorithm is $O(n+m_1+m_2)$. If one of those polygons is non-convex, we need to apply the results of the previous section. These results can be derived by a straightforward concatenation of the previous results. We let the details to the reader. We sum up the complexity results in the following table: (C stands for convex and NC for non-convex):

| E | $I_1$ | $I_2$ | Complexity |
|---|---|---|---|
| C | C | C | $O(n+m_1+m_2)$ |
| C | C | NC | $O((n+m_1m_2)\log(n+m_1m_2))$ |
| C | NC | C | idem |
| C | NC | NC | $O((n+(m_1m_2)^2)\log(n+m_1m_2))$ |
| NC | C | C | $n^4m_1{}^2m_2{}^2 \log nm_1m_2$ |
| NC | C | NC | $n^6m_1{}^3m_2{}^5 \log nm_1m_2$ |
| NC | NC | C | $n^6m_1{}^5m_2{}^3 \log nm_1m_2$ |
| NC | NC | NC | $n^8m_1{}^6m_2{}^6 \log nm_1m_2$ |

**Table 1**

The above algorithm provides all the valid relative positions, i.e. the valid relative positions between $I_1$ and $I_2$. These results are very pessimistics because they suppose that the number of intersections between the objects is always equal to the product of the numbers of their sides. Presumably, this number is rather proportional, in most practical situations, to the sum of the numbers of their sides than to their product. Under this hypothesis, the complexity of the NC NC NC case becomes $(n^2 m_1 m_2)\log(n m_1 m_2)$.

However the worst-case complexity can be significantly reduced if we are only interested in finding one valid relative position. In that case, a better algorithm can be obtained by decomposing $S_1$ and $S_2$ into convex parts and making use of Proposition 5. The detection algorithm is the following:

**Algorithm Double containment detection**

1- Compute $S_1=C(I_1,E)$ and decompose it into convex parts, $S_{11}...S_{1k}$;
2- Compute $S_2=C(I_2,E)$ and decompose it into convex parts, $S_{21}...S_{2l}$;
3- **If** $S_1$ or $S_2$ is empty **then return** "no valid relative position"
**else**
**3-1** Compute $I=O(I_2,I_1)$;
**3-2 For** j=1...l **do**
**3-2-1** Compute $S_j=O(S_{2j}, \cup_i S_{1i})$;
**3-2-2 If** $S_j^*$ and I intersect **then return** an intersection point;
**3-2-3 If** $S_j^* \supset I$ **then return** a vertex of $S_j^*$;
**3-3 Return** "no valid relative position";
4- **End.**

Instead of computing S, we compute the $S_j = O(S_{2j}, \cup_i S_{1i})$. The set of valid placements is given by $U = \cup_j (S_j^* - I)$. There exists a valid relative placement iff one of the $S_j$ intersects I or one of the $S_j$ contains I. Instead of computing the actual intersection between $S_j$ and I, we simply test if one exists and, if so, we report it. Whether any two of N line segments intersect can be determined in O(NlogN) time, and this is optimal (PS). An intersection is a valid relative position that corresponds to a placement where $I_1$ and $I_2$ touch the boundary of E and touch each other. In case that no intersection is found, we simply need to check if a vertex of $S_j^*$ exists which does not belong to the interior of I. If this is true, $S_j^*$ contains I and that vertex is a valid relative position; otherwise there is no valid relative position. This detection algorithm is faster than the previous one when E is non-convex. We sum up below the complexity results; m stands for $\min(m_1,m_2)$ (see appendix A for details):

| E | $l_1$ | $l_2$ | Complexity |
|---|---|---|---|
| NC | C | C | $n^2m_1m_2 \log nm_1 \log nm_2$ |
| NC | C | NC | $n^3m_1m_2^2 \log nm_1 \log nm_2$ |
| NC | NC | C | $n^3m_1^2m_2 \log nm_1 \log nm_2$ |
| NC | NC | NC | $n^4m_1^2m_2^2 \log nm_1 \log nm_2$ + $n^2m_1^2m_2^2\, m^2 \log nm_1m_2$ |

**Table 2**

Once a relative position has been found, it remains to determine a placement.

**Theorem 7:** *For a given relative position u, the set P(u) of all the possible placements is given by* $P(u) = S_1 \cap T_{-u}(S_2)$.

**proof:** lemma 2 of section 3 in the case of two polygons. ♦

Computing P(u) does not take more time than computing the relative positions, so the overall complexity of finding a valid placement for $l_1$ and $l_2$ is given by the tables above.

## 5- SIMULTANEOUS CONTAINMENT OF THREE POLYGONS

We consider now three polygons $l_1$, $l_2$ , $l_3$ and give a solution to the simultaneous containment problem in case that polygon E is a parallelogram .

A relative position of $l_1$, $l_2$, $l_3$ is given by a couple of vectors $(u_2,u_3)$ where $u_2$ is a relative position of $(l_1,l_2)$ and $u_3$ is a relative position of $(l_1,l_3)$ . Recall that a relative position $(u_2,u_3)$ is valid iff there exists a placement with such a relative position . As in section 3 the method consists in two steps : at first , we compute the valid relative positions of $l_1$, $l_2$, $l_3$ and then, for a given relative position, say $(u_2,u_3)$ , we compute the set $P(u_2,u_3)$ of all the possible placements.

Let $U_{jk}$ be the set of valid relative positions for $(l_j,l_k)$ and U the set of the valid relative positions $(u_2,u_3)$ of $l_1$, $l_2$, $l_3$ .

Let us note $U_2 = \{ u_2 / \exists u_3$ such that $(u_2,u_3) \in U \}$ and $U_3(u_2) = \{ u_3 / (u_2,u_3) \in U \}$

**Theorem 8** : $U_2 = U_{12} \cap O(U_{23}, U_{13})$.

**proof**: (see fig. 8) Let $u_2$ be a vector of $U_2$. There exists $u_3$ such that $(u_2,u_3)$ belongs to U. As $(u_2,u_3)$ is a valid relative position for $I_1$, $I_2$ , $I_3$ $u_2$ (resp. $u_3$) is necessarily a valid relative position of $(I_1, I_2)$ (resp. $(I_1, I_3)$) . So $u_2$ (resp. $u_3$) belongs to $U_{12}$ (resp. $U_{13}$). Let us note $v=u_3-u_2$; v is necessarily a valid relative position of $(I_2, I_3)$ so v belongs to $U_{23}$. We have $u_2=u_3-v$ and then $u_2$ belongs to the Minkowski difference between $U_{13}$ and $U_{23}$ which is exactly $O(U_{23}, U_{13})$.

Conversely, let $u_2$ be a vector of $U_{12} \cap O(U_{23} , U_{13})$. $u_2$ belongs to $U_{12}$ and there exists $u_3$ in $U_{13}$ and v in $U_{23}$ such that $u_2=u_3-v$; consider now the relative position $(u_2,u_3)$ for $I_1$, $I_2$ , $I_3$. In that relative position $I_1$, $I_2$ , $I_3$ do not overlap. Let us note $S_j$ j=1,2,3 the set of all the valid placements for $I_j$ j=1,2,3 in E. $S_j$ is a parallelogram, a segment or a point with edges parallel to the edges of E (see appendix B). Moreover, because $u_2$ is in $U_{12}$, $u_3$ is in $U_{13}$ and v is in $U_{23}$ we have :

$$S_1 \cap T_{-u2}(S_2) \neq \emptyset \qquad (1)$$

$$S_1 \cap T_{-u3}(S_3) \neq \emptyset \qquad (2)$$

$$S_2 \cap T_{-v}(S_3) \neq \emptyset \qquad (3)$$

From (3) we deduce $T_{-u2}(S_2 \cap T_{-v}(S_3))=T_{-u2}(S_2) \cap T_{-u3}(S_3) \neq \emptyset$ . We conclude that the three sets $S_1$, $T_{-u2}(S_2)$ , $T_{-u3}(S_3)$ are parallelograms that intersects two by two , thus $S_1 \cap T_{-u2}(S_2) \cap T_{-u3}(S_3) \neq \emptyset$ (see appendix B) hence $u_2$ is in $U_2$ and any vector in this intersection is a placement for the relative position $(u_2,u_3)$. ♦

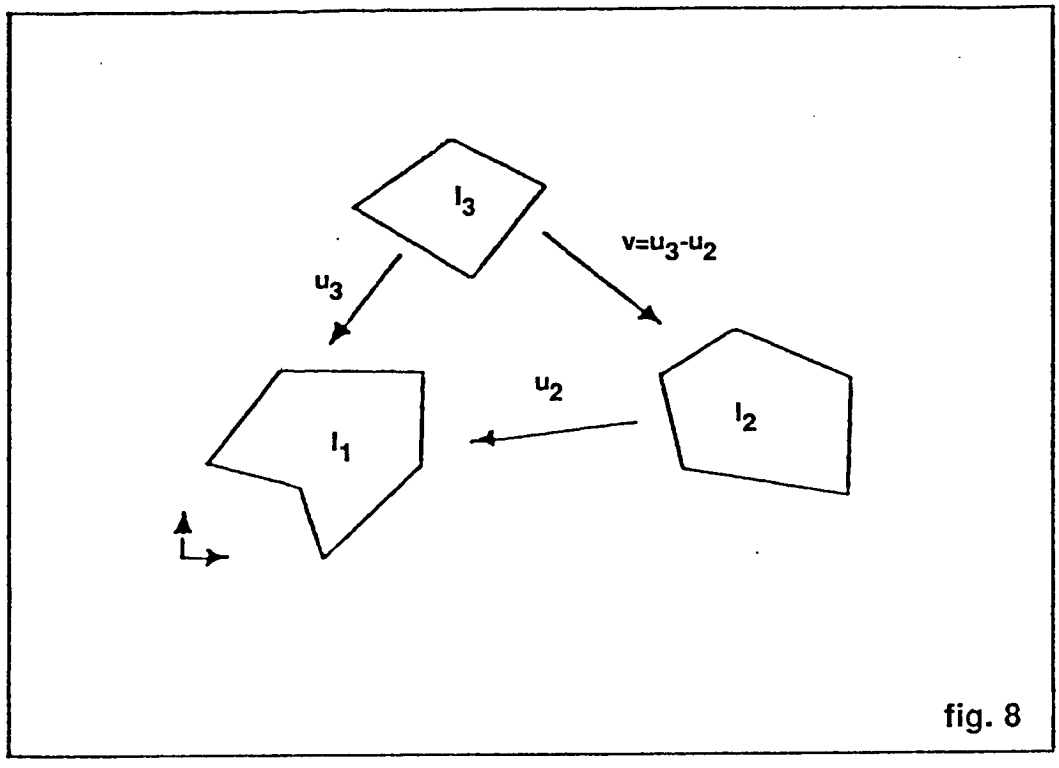


fig. 8

If we choose now a vector $u_2$ in $U_2$ (assuming that $U_2$ is not empty) we have to find the set $U_3(u_2)$ of all valid relative positions of $I_1$, $I_2$, $I_3$ with first vector $u_2$:

**Theorem 9**: $U_3(u_2) = O(S_3, S)^* - (O(I_3,I_1) \cup T_{u2}(O(I_3, I_2))$ *with* $S=T_{-u2}(S_2) \cap S_1$

**proof**: Let $u_3$ be in $O(S_3,S)^* - (O(I_3, I_1) \cup T_{u2}(O(I_3, I_2)))$. $u_3$ is in $O(S,S_3)^*$ implies that $T_{-u3}(S_3) \cap S = T_{-u3}(S_3) \cap T_{-u2}(S_2) \cap S_1 \neq \emptyset$ .

Let us show now that $I_1$, $T_{u2}(I_2)$ and $T_{u3}(I_3)$ do not overlap: by hypothesis, $I_1$ and $T_{u2}(I_2)$ do not overlap; $u_3$ is not in $O(I_3, I_1)$ so $I_1$ and $T_{u3}(I_3)$ do not overlap. $u_3$ is not in $T_{u2}(O(I_3, I_2))$ so $u_3-u_2$ is not in $O(I_3, I_2)$ then $T_{u3-u2}(I_3)$ and $I_2$ do not overlap and so do $T_{u3}(I_3)$ and $T_{u2}(I_2)$.

Suppose that $u_3$ is in $U_3(u_2)$; $(u_2,u_3)$ is a valid relative position of $I_1$, $I_2$ $I_3$. Necessariiy, $S_1 \cap T_{-u2}(S_2) \cap T_{-u3}(S_3)$ is not empty which is equivalent to $u_3$ belongs to $O(S_3, S)^*$. Moreover, $T_{u3}(I_3)$ and $I_1$ do not overlap then $u_3$ is not in $O(I_3,I_1)$, $T_{u3}(I_3)$ and $T_{u2}(I_2)$ do not overlap then $u_3$ is not in $T_{u2}(O(I_3, I_2))$. ♦

Given a valid relative position of U, say $(u_2, u_3)$, we can find the set $P(u_2,u_3)$ of all the placements corresponding to that valid relative position:

**Theorem10**: $P(u_2,u_3)=S_1 \cap T_{-u2}(S_2) \cap T_{-u3}(S_3)$.

**proof**: lemma 2 of section 3 in the case of three polygons. ♦

These three theorems yield an algorithm which first exhibits a valid relative position $(u_2,u_3)$ and then computes $P(u_2,u_3)$. This algorithm is described below:

**Algorithm Triple parallelogram containment**

1-/* Computation of $U_2$*/

**1.1** Compute $U_{12}$, $U_{23}$, $U_{13}$; **If** one of them is empty **then** return $\emptyset$;

**1.2** Compute $O=O(U_{23}, U_{13})$;

**1.3** Compute $U_2=U_{12} - O$;

2- **If** $U_2=\emptyset$ **then** return $\emptyset$

**else** choose a vector, say $u_2$, in $U_2$ and **go to** step 3;

3- /* Computation of $U_3(u_2)$*/

**3.1** Compute $S=T_{-u2}(S_2) \cap S_1$;

**3.2** Compute $O(S_3, S)$ then $O(S_3, S)^*$;

**3.3** Compute $O=O(I_3, I_1) \cup T_{u2}(O(I_3, I_2))$;

**3.4** Compute $U_3(u_2)= O(S, S_3)^* - O$;

4- Choose a vector, say $u_3$, in $U_3(u_2)$;

5- **Return** $P(u_2,u_3)=S_1 \cap T_{-u2}(S_2) \cap T_{-u3}(S_3)$;

6- **End.**

This algorithm works for convex or non-convex polygons $I_1$, $I_2$, $I_3$. Using the previous results, the complexity analysis is quite straightforward.

## 6-THREE DIMENSIONAL GENERALISATION

Our methods can be generalized to higher dimensions and used to solve the containment problem for non convex polytopes. Notice that Theorems 3, 6, 7 hold when I, $I_1$, $I_2$, E are non convex polytopes in d-dimensional Euclidean space; Theorems 8, 9, 10 hold when $I_1$, $I_2$, $I_3$ are non convex polytopes and E is a parallelepiped in d-dimensional Euclidean space. In order to make use of these theorems for an effective computation of the feasible region, we need algorithms to compute C(I,E), O(A,B) when I,E,A,B are convex polytopes, to decompose a non convex polytope into convex parts and to perform boolean operations on non convex polytopes. We show below that these basic algorithms are available in the 3-dimensionnal case. Notice that the complexity of the problem is then increased significantly as shown in fig. 9 where the feasible region for the case of two polyhedra has $\Omega(n^3m^3)$ edges (S).

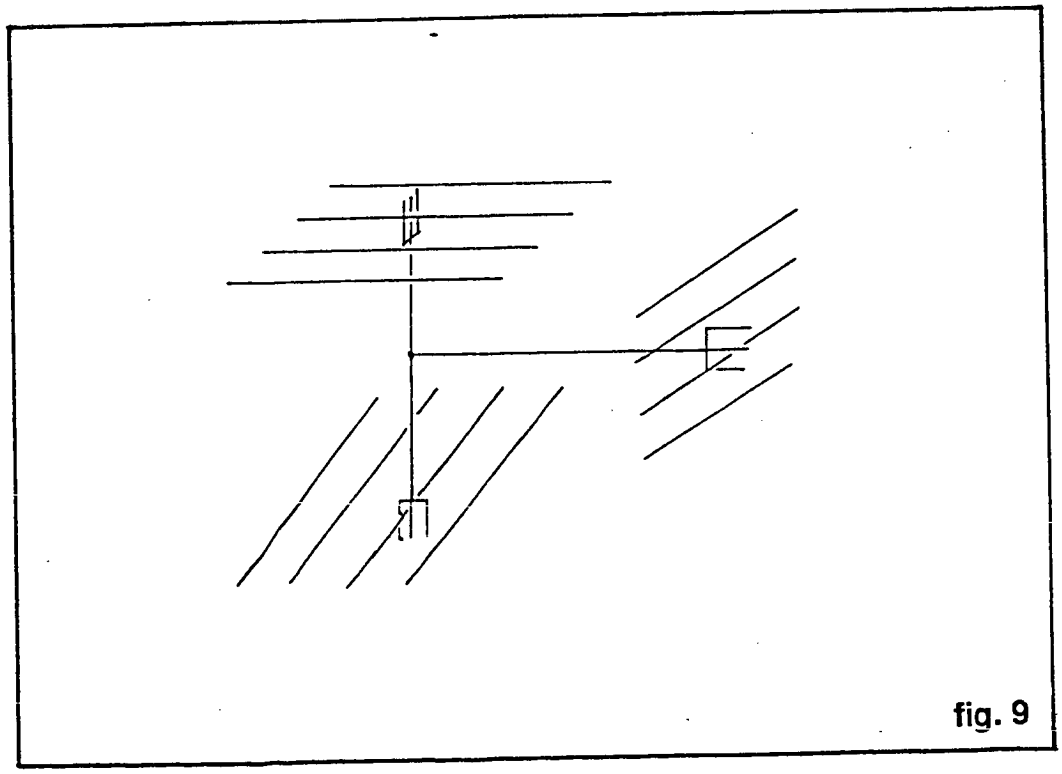

fig. 9

## 6-1 Containment problem for a d- (convex) polytope in d-dimensional Euclidean space

Let E and I be d-polytopes with vertices $E_1,..., E_n$ and $I_1,..., I_m$ respectively. Suppose for simplicity that $I_1$ is at the origin of the reference frame. Let $T_j$ be the translation of vector $-\mathbf{I_1I_j}$ and C(I,E) be the set of translations that fit I inside E.

**Theorem 11**: $C(I,E) = \cap_j T_j(E)$.

**proof**: (see fig. 10, fig. 11) Let u be a vector of C(I,E); for any $j \in \{1 .. m\}$
$u = (\mathbf{I_1I_j} + u) - \mathbf{I_1I_j}$ where $\mathbf{I_1I_j} + u$ is a vector of E (j=5 in fig. 10).
Let u be a vector of $\cap_j T_j(E)$; for $j \in \{1 .. m\}$ $T_u(I_j) = u + \mathbf{I_1I_j}$ is in E (j=5 in fig. 11). Indeed, u is in $T_j(E)$ thus can be written as $e - \mathbf{I_1I_j}$ where e is in E. As E is convex, we deduce that the convex hull of the set of points $\{T_u(I_j)\}$ which is $T_u(I)$ (I is convex) is in E. ♦

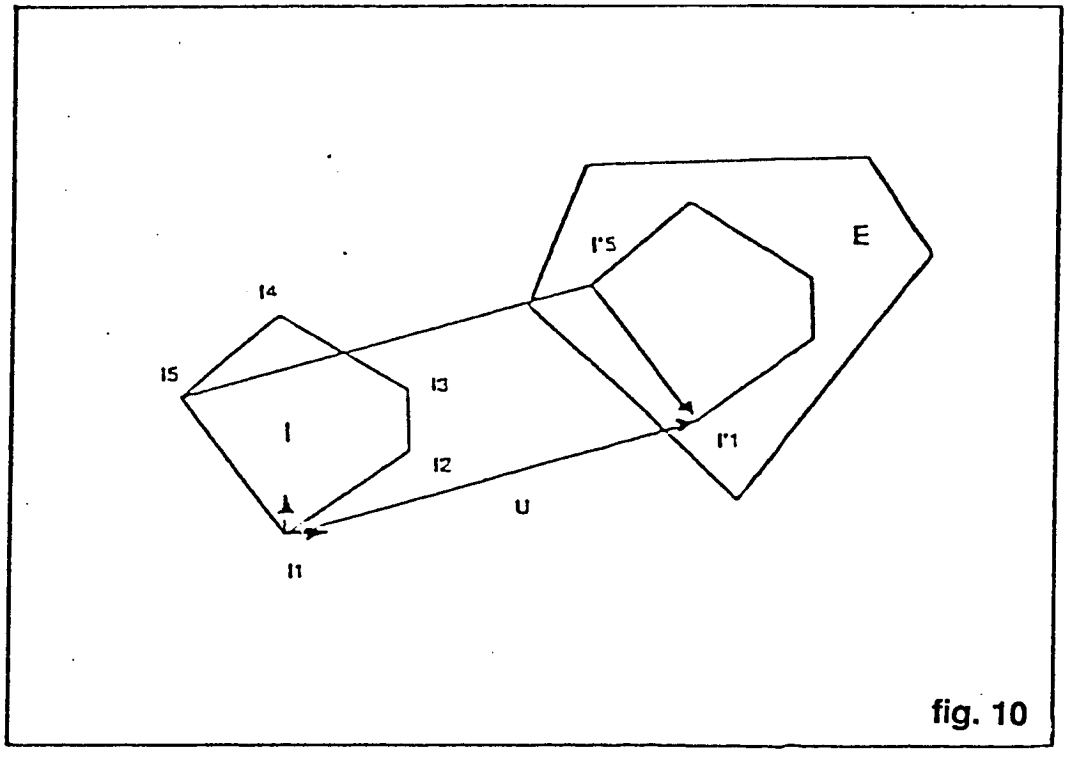


fig. 10

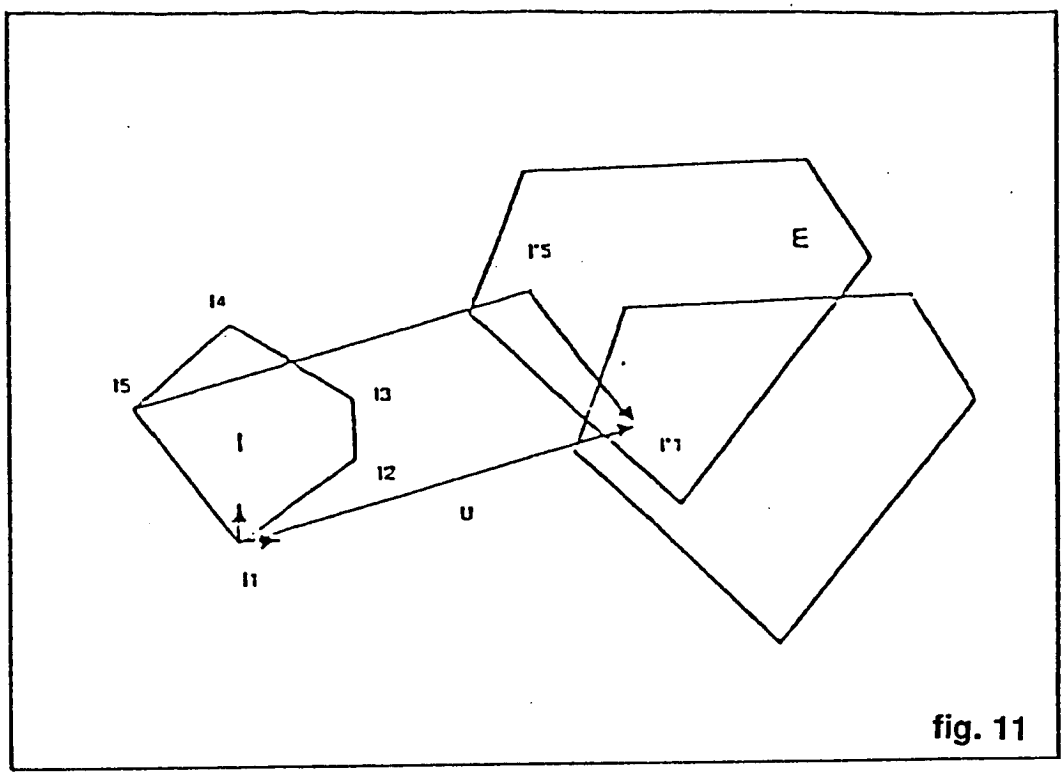


fig. 11

**Corrolary:** *C(I,E) is a convex d'-polytope with $0 \leq d' \leq d$.*

### 6-2 Minkowsky difference between d-(convex) polytopes in d-dimensional Euclidean space

Let A and B be d-polytopes with vertices $A_1, .., A_n$ and $B_1, .., B_m$ respectively. Suppose for simplicity that $B_1$ is at the origin of the reference frame. Let $T_j$ be the translation of vector $-\mathbf{B_1B_j}$ and O(A,B) be the set of translations that make B to intersect A.

We note $L = \{ A_k - \mathbf{B_1B_j},\ k \in \{1 .. n\}, j \in \{1 .. m\} \}$.

**Theorem 12:** *O(A,B) = CH(L).*

**proof:** (see fig. 12) Let u be a vector of O(A,B) and M a point in $A \cap T_u(B)$. Let us note $J=T_{B1M}(B^*)$, $B_i'=T_u(B_i)$ and $v=B_1'M$. As usual, B* denotes the polygon which is symetric to B with respect to the origin. We prove first that CH(L) contains O(A,B). As M is in $T_u(I)$ , it is clear that u belongs to J. Indeed, $J=T_v(\text{sym } T_u(B)/B_1')$ and thus $B_1'$ is in J. Moreover, as M is in A, all the vertices of J belong to CH(L) and, by convexity, J is in CH(L). Conversely, O(A,B) is the Minkowski difference between A and B which is known to be convex. Moreover, it is clear that all the points of L are in O(I,E) so O(I,E) contains CH(L). ♦

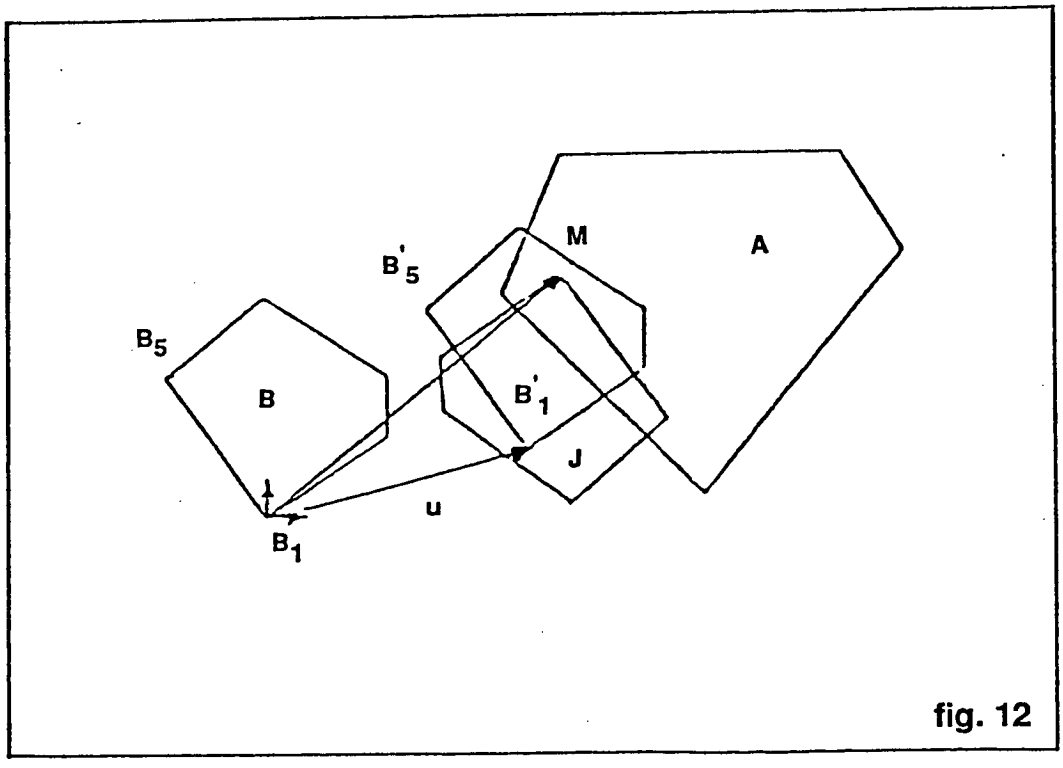


fig. 12

**Corrolary:** *O(A,B) is a convex d-polytope*

### 6-3 Algorithmic issues

In order to apply the same method as for the 2-d case, we need at first to be able to compute C(I,E) and O(I,E) in the case where I and E are both convex polyhedra.

Theorem 11 yields an algorithm to compute C(I,E). Indeed, C(I,E) is the common intersection of mn half-planes which can be computed in time O(mn log mn) in the 3-d case (PS).

Theorem 12 yields an algorithm to compute O(I,E). Indeed, O(I,E) is the convex hull of mn points which can be computed in time O(mnlogmn). This is close to optimal in the worst-case since O(I,E) may have O(mn) edges. A much more involved technique due to Guibas and Seidel (GS) computes O(I,E) in time O(n+m+k) where k is the size of the output which may be O(mn).

In order to solve the general 3-d case, we need also to decompose a polyhedron with n vertices into convex parts and to compute Boolean operations on polyhedra. The first problem can be solved by triangulating the polyhedron. A way to do that consists in computing the Delaunay triangulation of the vertices of the polyhedron added with some other points taken on the edges of the polyhedron in order to guarantee the polyhedron's faces to be contained in the triangulation. This can be done in $O(n^2)$ time (B). The result is a set of $O(n^2)$ tetrahedra in the worst-case. A challenging problem would be to try to reduce these bounds. Computing the intersection and the union of a convex polyhedron and a concave one can be performed by space-sweep techniques (HMM,MS) .

Thus there exist rather efficient algorithms for all the operations relevant to the problem of fitting one or two simple concave polyhedra inside another one and fitting three concave simple polyhedra into a parallelepiped without overlapping .

## 7-IMPLEMENTATION

These algorithms have been implemented in C on a SUN workstation. The system is mainly based on the implementation of three classes of primitives which are described below:

**1**- The first class computes C(P,Q) and O(P,Q) where P and Q are convex; the implementation uses of two linear-time algorithms based on Propositions 11 and 12;

**2**- The second class computes a decomposition of a polygon into convex parts and the convex-hull of a polygon; they have been directly implemented, without difficulty, from the detailed algorithms given in (HM).

**3**- Boolean operations on family of polygons have been implemented from the algorithm given in (OWW).

The last two classes of primitives are based on plane sweep technique and require the implementation of data structures which supports the MIN, INSERT, DELETE, SUCC and PRED operations: we use binary search trees of bounded balance. (NR)

The boolean operations algorithm given in (OWW) works for families of polygons with no vertex in common, nor edges in common nor portions of edges in common. Unfortunately, the case we are working on does not accept such restrictions. Indeed, when we decompose polygons into convex parts, some of these convex parts share common edges and/or common vertices. So the Minkowsky differences between families of such polygons yield families of convex polygons with common vertices, common edges and, in some cases, common portions of edges. We have extended the algorithm to that case and written an algorithm which solves the general case.

Experimental results of our implementation are shown in fig. 13 a,b,c 14 d,e,f,g 15 h,i,j. Polygons are in solid lines, holes are hatched. Results are in dashed lines, and are given for a reference point, surrounded by a circle. Polygons to be translated are on the right on the figure, and a placement is shown on the left of the figure.

The most costly steps in all the containment algorithms we have presented are the steps computing boolean operations on polygons. Indeed, the complexity of the algorithm depends on the number of intersection points between the polygons of the families we have to process and this number can be very large compared to the number of edges of the result.

A way to significantly reduce the computing time would be the use of the graphics tools and the associated specific hardware now available on most workstations equipped with a bit-map. Computing boolean operations on polygons can be performed by 1/ painting the polygons with appropriate colours, thus replacing the polygons by sets of coloured pixels; 2/ computing the desired boolean operations at each pixel of the bit-map; 3/ extracting the boundary of the solution.

## 7- CONCLUSION

A general solution to the containment problem by translation for one and two polygons and a partial solution for the case of three polygons have been given. The different cases have been treated in a uniform way which allow us to only make use of a small set of primitives that can be implemented rather easily. Some algorithms only find one feasible placement if one exists. Others find an implicit representation of the *whole* feasible region. The methods extend easily to the 3-dimensional case, although the complexity of the algorithms and the difficulty to code them increase significantly.

Further research on this subject includes solving the problem for more than three polygons and allowing rotations. Another interesting topic would be to apply the implicit representation introduced in this paper to other related problems and, in particular, to the coordinated motion of several polygonal objects.

## 8- APPENDIX A

We note IPI the number of edges of P. $S_1$ and $S_2$ are decomposed into, respectively, k and l convex parts $S_{11}...S_{1k}$, $S_{21}...S_{2l}$. We have $S_j=O(S_{2j}, \cup_i S_{1i})$. From Proposition 5, $S_j$ has $|S_j|=O(|S_1|+k|S_{2j}|)$ edges and can be computed in $|S_j| \log |S_j| \log k$ time. We can test if $S_j$ and I intersect in $O((|I|+|S_j|) \log(|I|+|S_j|)$ time, except in the case where I and $S_j$ are convex; the complexity is then $O((|I|+|S_j|)$.

The total complexity of step 3 of the algorithm and thus the overall complexity is given by:

$$T_1 = \Sigma_{j=1,l}\ O\ (\ |S_j| \log |S_j| \log k\ +\ (|I|+|S_j|)\ \log(|I|+|S_j|)\ )$$
$$= \Sigma_{j=1,l}\ O\ ((|S_1|+k|S_{2j}|) \log\ (\ |S_1|+k|S_{2j}|)\ \log k\ +\ (|I|+\ |S_1|\ +k\ |S_{2j}|)\log\ (|I|+\ |S_1|\ +k\ |S_{2j}|))$$

In the worst case $k=O(|S_2|)$ and $l=O(|S_1|)$, thus

$$T_1 \leq |S_1|\,|S_2| \log |S_1|\,|S_2| \log|S_2| + (|||\,|S_2| + |S_1|\,|S_2|) \log (|||\, + |S_1|\,|S_2|)).$$

Of course, a similar result holds if we exchange the roles of $l_1$ and $l_2$, yielding a complexity $T_2$. Thus the actual complexity is $T=\min(T_1, T_2) \leq |S_1|\,|S_2| \log |S_1| \log|S_2| + (||| |S|+ |S_1|\,|S_2|) \log (|||\, + |S_1|\,|S_2|))$, with $|S|=\min(|S_1|,|S_2|)$. The values of $|S_1|$, $|S_2|$ and $|||$ are given in the following table:

| E | $l_1$ | $l_2$ | $\|S_1\|$ | $\|S_2\|$ | $\|\|\|$ |
|---|---|---|---|---|---|
| NC | C | C | $nm_1$ | $nm_2$ | $m_1+m_2$ |
| NC | C | NC | $nm_1$ | $(nm_2)^2$ | $m_1m_2$ |
| NC | NC | C | $(nm_1)^2$ | $nm_2$ | $m_1m_2$ |
| NC | NC | NC | $(nm_1)^2$ | $(nm_2)^2$ | $(m_1m_2)^2$ |

**Table 3**

In all cases, we have $|||\leq|S_1||S_2|$ and, apart from the NC NC NC case, $|||S|\leq|S_1||S_2|$. Thus

$$T\leq |S_1|\,|S_2| \log |S_1| \log|S_2| + ||| |S|\log (|S_1|\,|S_2|).$$

The results given in table 2 can be easily deduced from this formula.

## 9- APPENDIX B

**Proposition:** *If I is a polygon and P a parallelogram, C(I,P) is empty or a point, a segment a parallelogram.*

**proof:** a direct consequence of theorem 11. ♦

**Proposition:** *If $P = \{P_1, .. , P_k\}$ is a collection of parallelograms with parallel edges,*
$\cap_i P_i \neq \varnothing$ *iff* $P_l \cap P_j \neq \varnothing \quad \forall l,j$

**proof:** the condition $P_i \cap P_j \neq \varnothing \ \forall$ i,j is necessary; Let us show that it is sufficient: consider a reference frame $(O,d_1,d_2)$ where $d_1,d_2$ are the two common directions of the edges of the $P_i$. Let us note $pr_1$(resp. $pr_2$) the first (resp. second) projection associated with reference frame $(O,d_1,d_2)$. We have $\cap_i P_i = pr_1(\cap_i P_i) \times pr_2(\cap_i P_i)$; Moreover, $pr_l(\cap_i P_i) = \cap_i pr_l(P_i)$ (l=1,2). As $pr_l(P_i)$ is a segment on the line $(O,d_l)$ (l=1,2) and these segments intersect two by two by hypothesis, Helly's theorem (MMS) in one dimension can be applied and we deduce that $pr_l(\cap_l P_l) = \cap_i pr_l(P_i) \neq \varnothing$ (l=1,2) and thus $\cap_i P_i \neq \varnothing$. ♦

**13.a.** containment of a convex polygon :

1. in a convex polygon : the feasible region is a convex polygon.
2. in the same polygon with a convex hole : the feasible region is not convex.

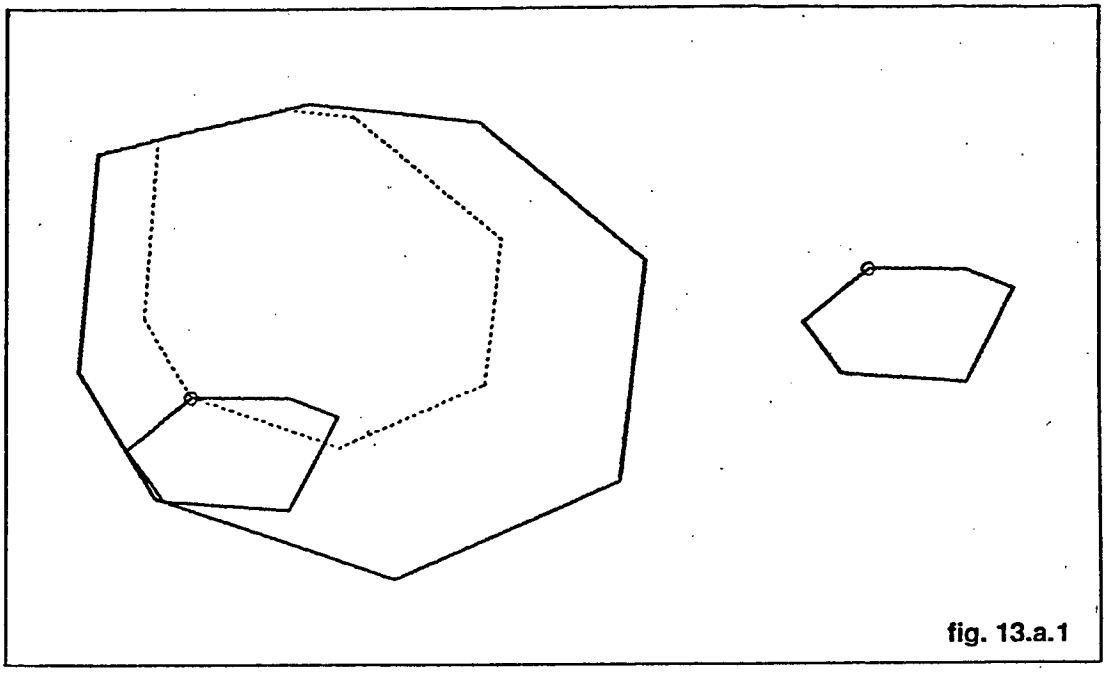

fig. 13.a.1

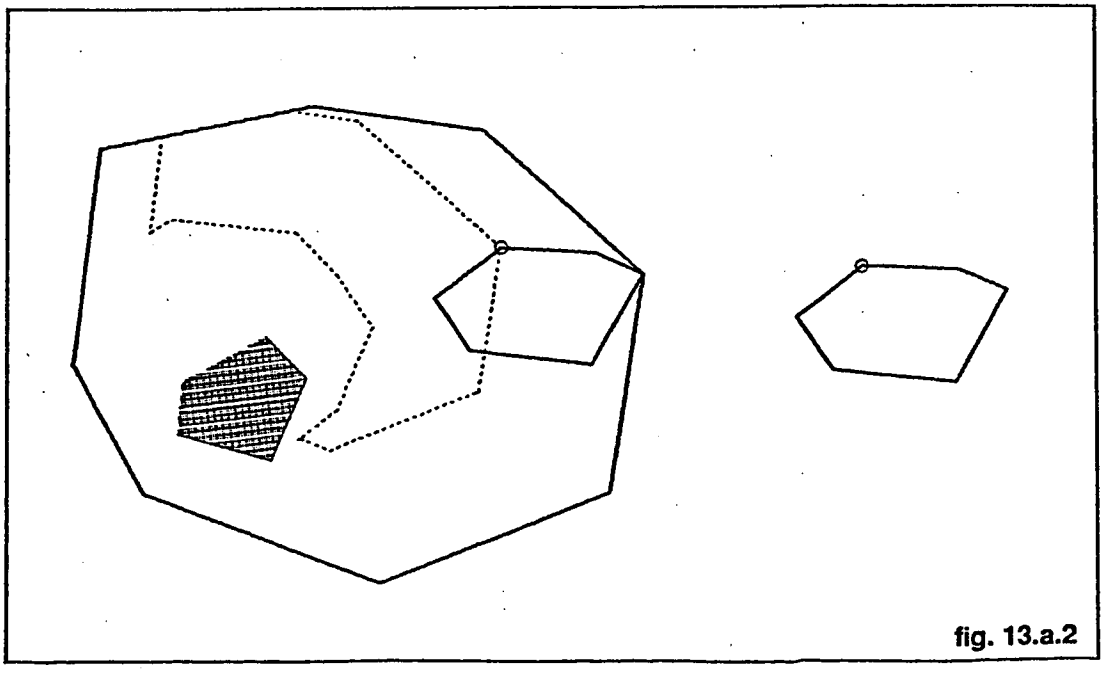

fig. 13.a.2

**13.b.** containment of a convex polygon :

1. in a non convex polygon : the feasible region is a polygonal region made of three regions.

2. in the same polygon with convex holes : the feasible region is still composed of three regions but one of them is not affected by holes.

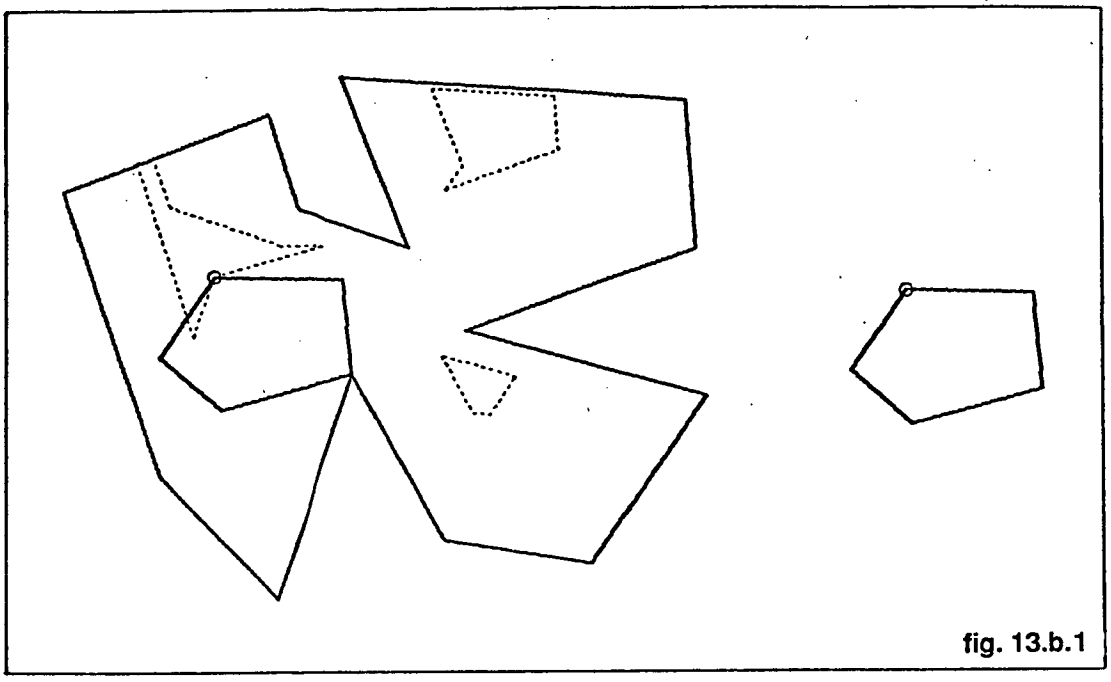

fig. 13.b.1

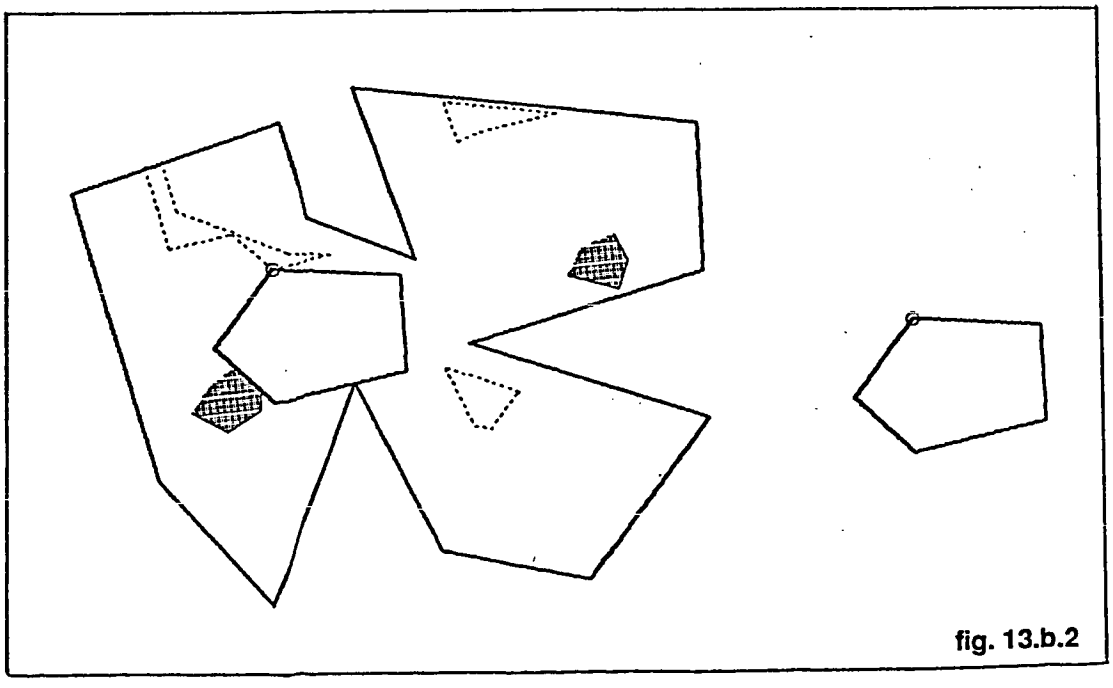

fig. 13.b.2

**13.c.** containment in a non convex polygon :

1. in a non convex polygon : the feasible region is a polygonal region composed of two polygons.

2. in the same polygon with non convex holes : the feasible region is now composed of three polygons.

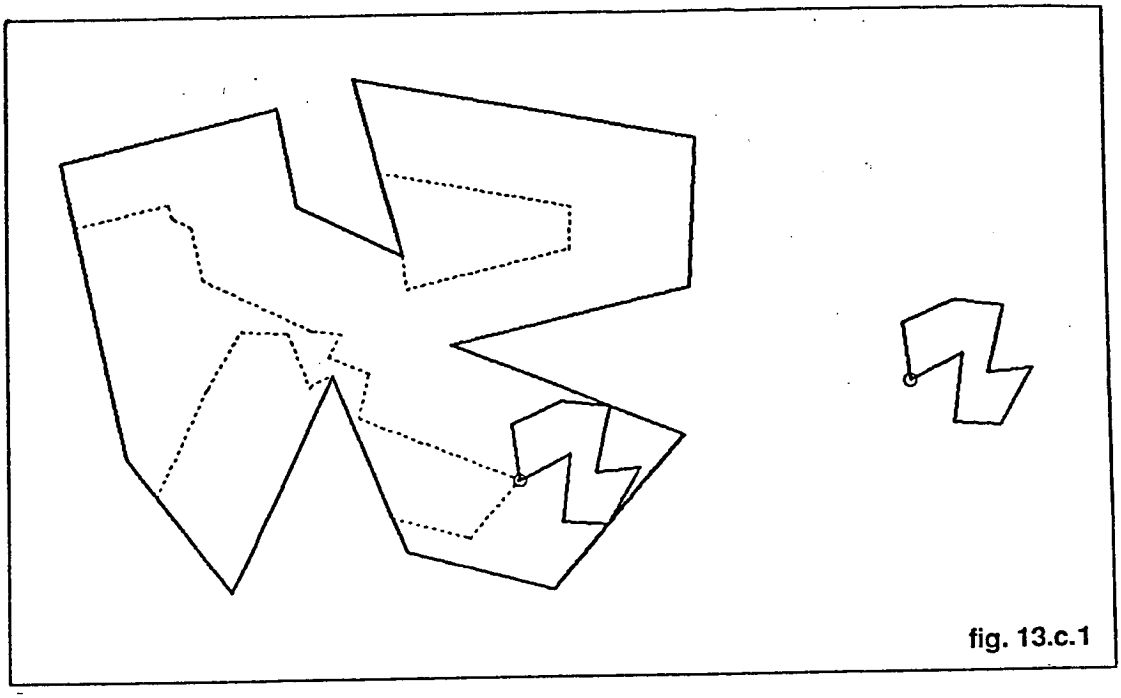

fig. 13.c.1

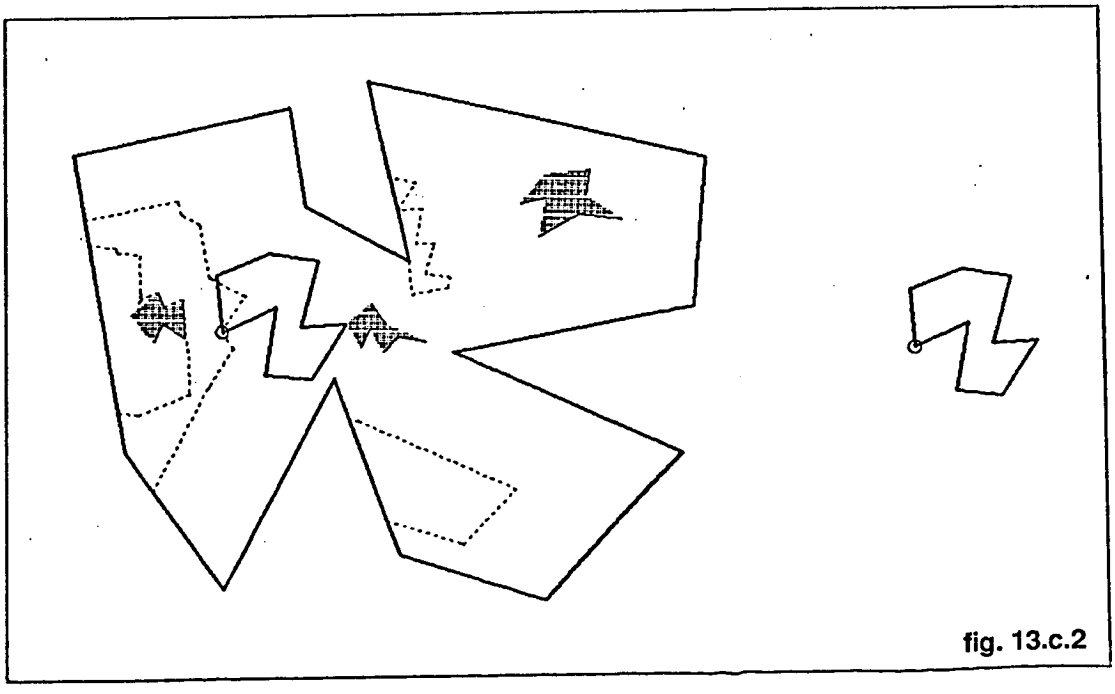

fig. 13.c.2

**14.d.** containment of two non convex polygons in a non convex polygon with a non convex hole :

The right dashed lines polygon is the set of the valid relative positions . A valid relative position beeing chosen (thin line circle reference point), the feasible region for the couple of polygons is exhibited (bold line circle reference point).

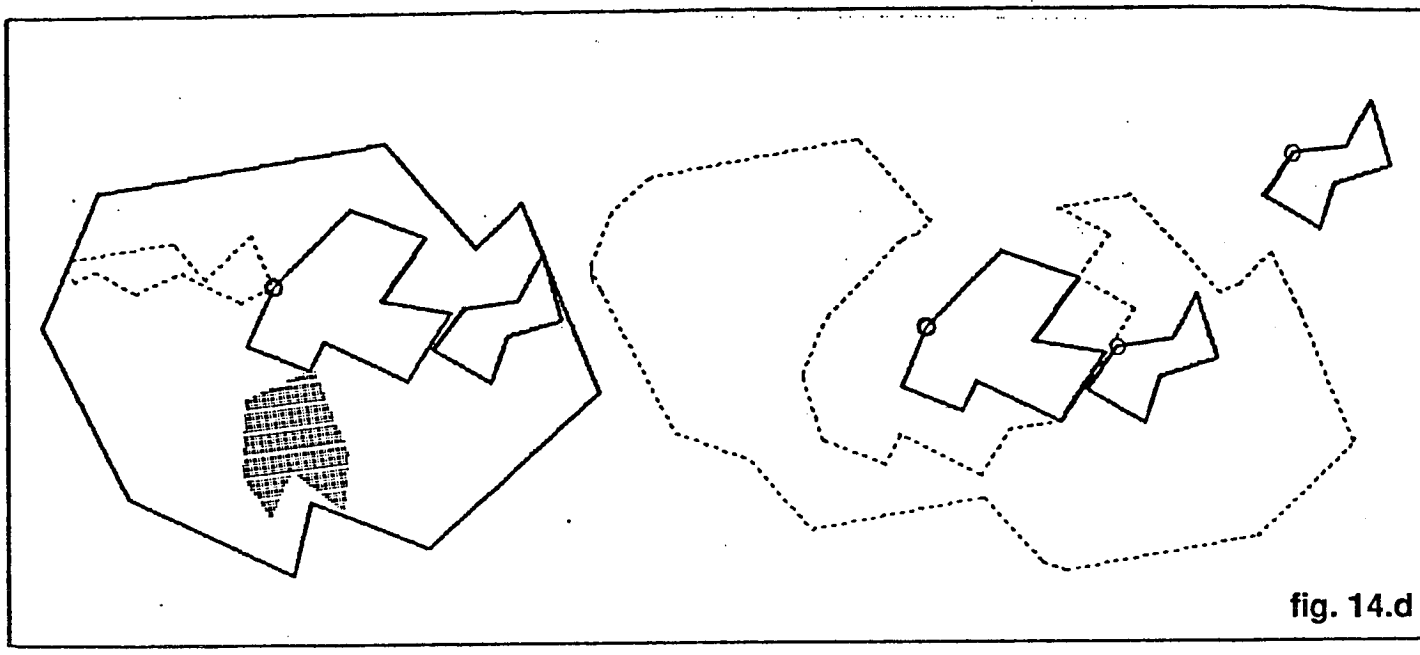

fig. 14.d

**14.e,f,g.** same as **14.d.** for a different choice of the relative position.

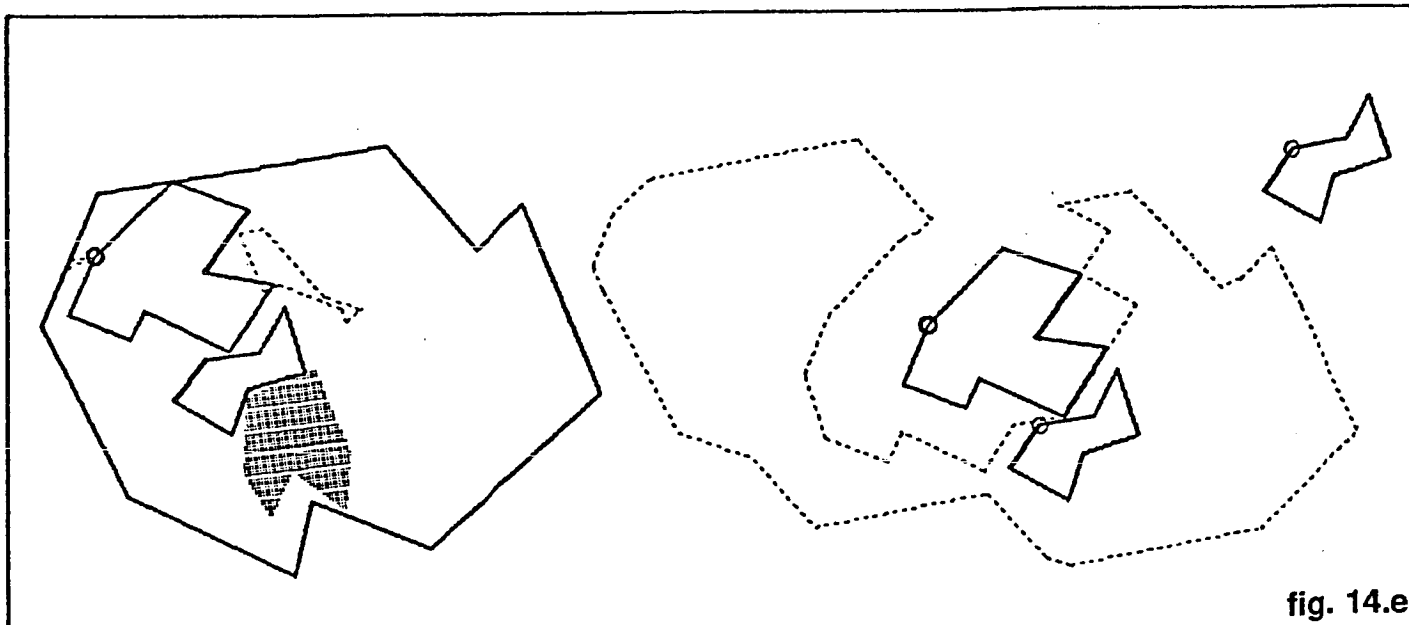

fig. 14.e

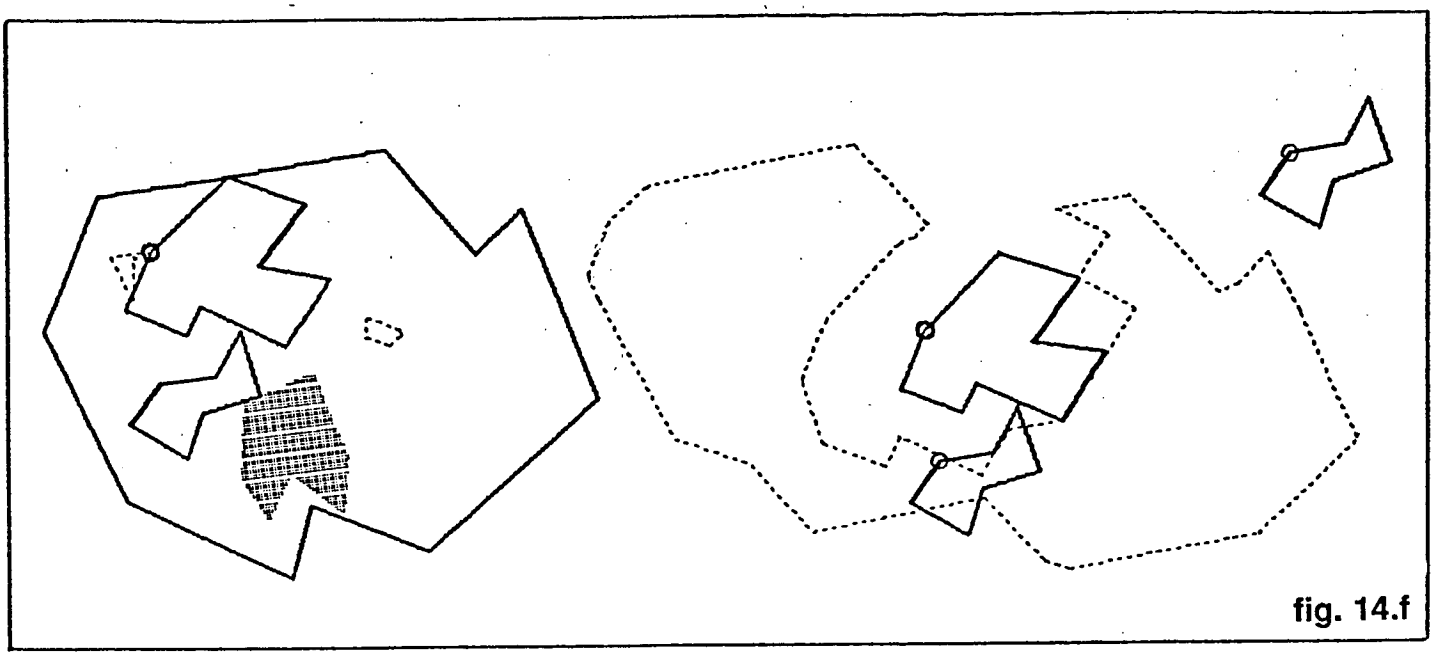

fig. 14.f

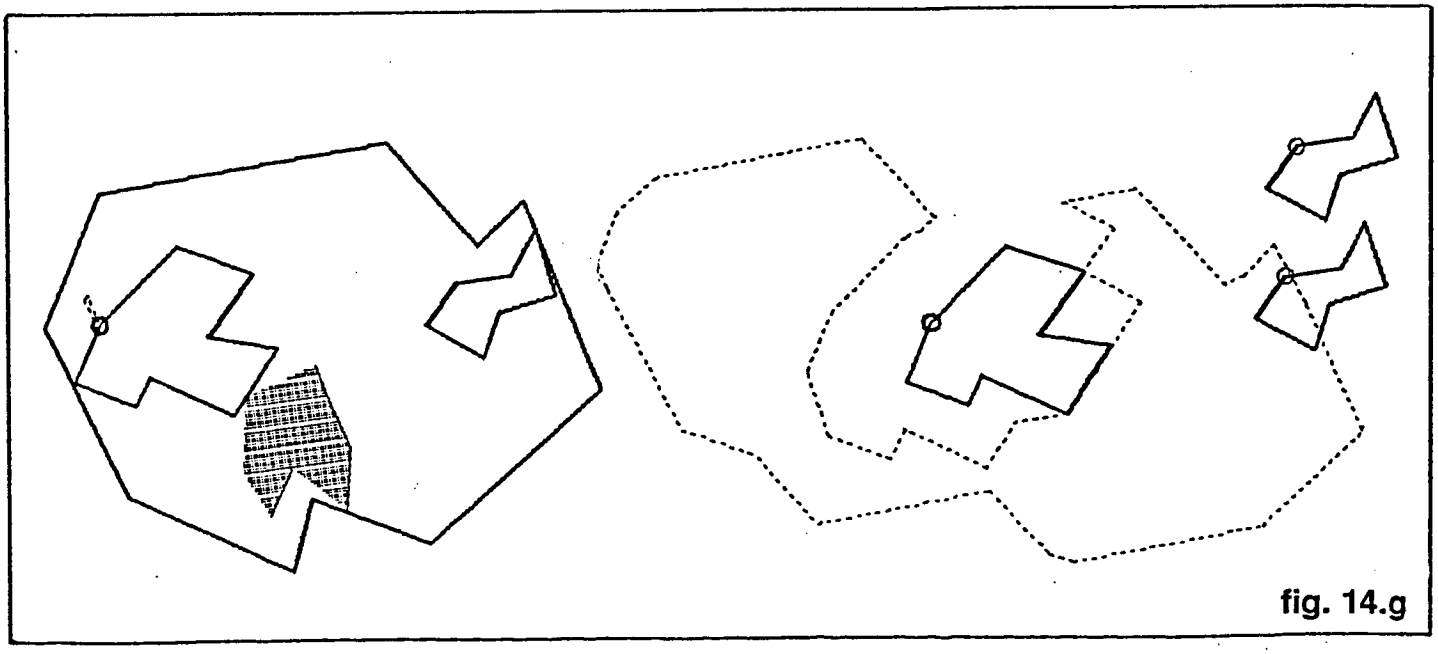

fig. 14.g

**15.h.** containment of three non convex polygons in a parallelogram :

The bold dashed lines polygon is the set of valid relative positions for the couple consisting of the rightmost polygon and the one in the middle. A valid relative position beeing chosen (bold line circle), the set of valid relative positions for the couple consisting of the leftmost polygon and the right couple is exhibited in thin dashed line. A valid relative position is chosen (thin line circle) and the feasible region for the three polygons is given in solid lines (it is a parallelogram).

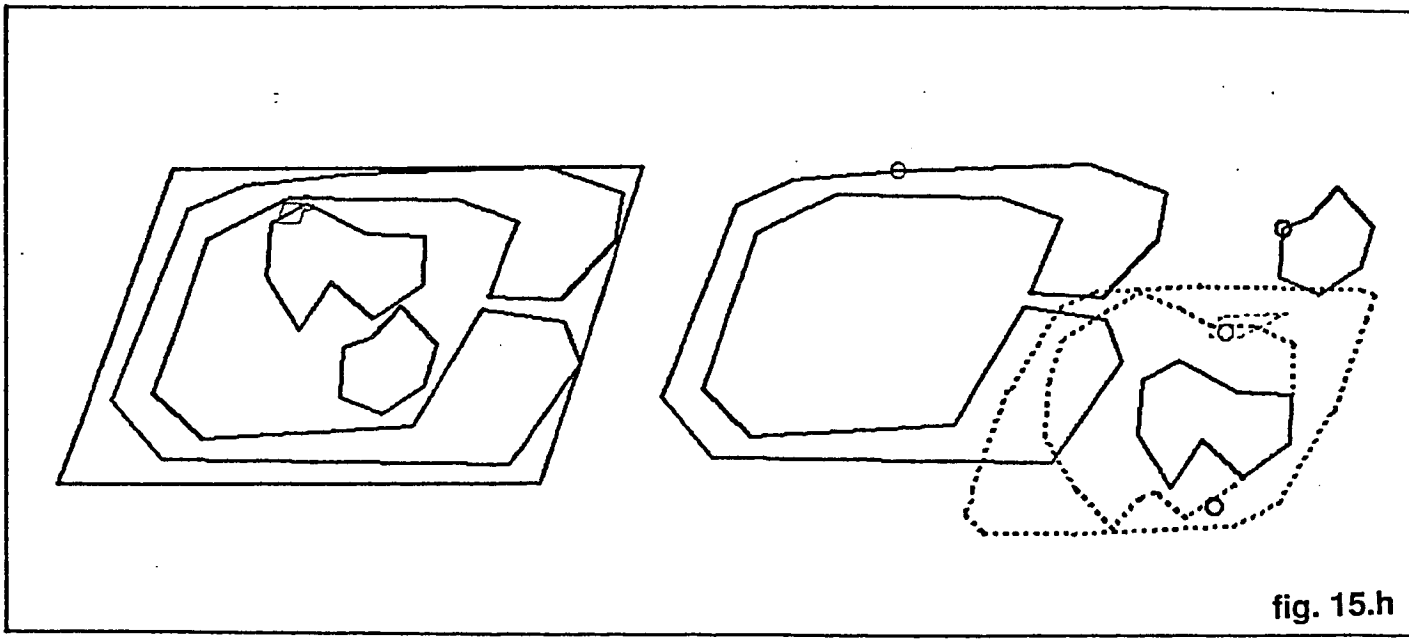

fig. 15.h

**15.i,j.** same as **15.h.** for differents polygons and differents choices of relative positions.

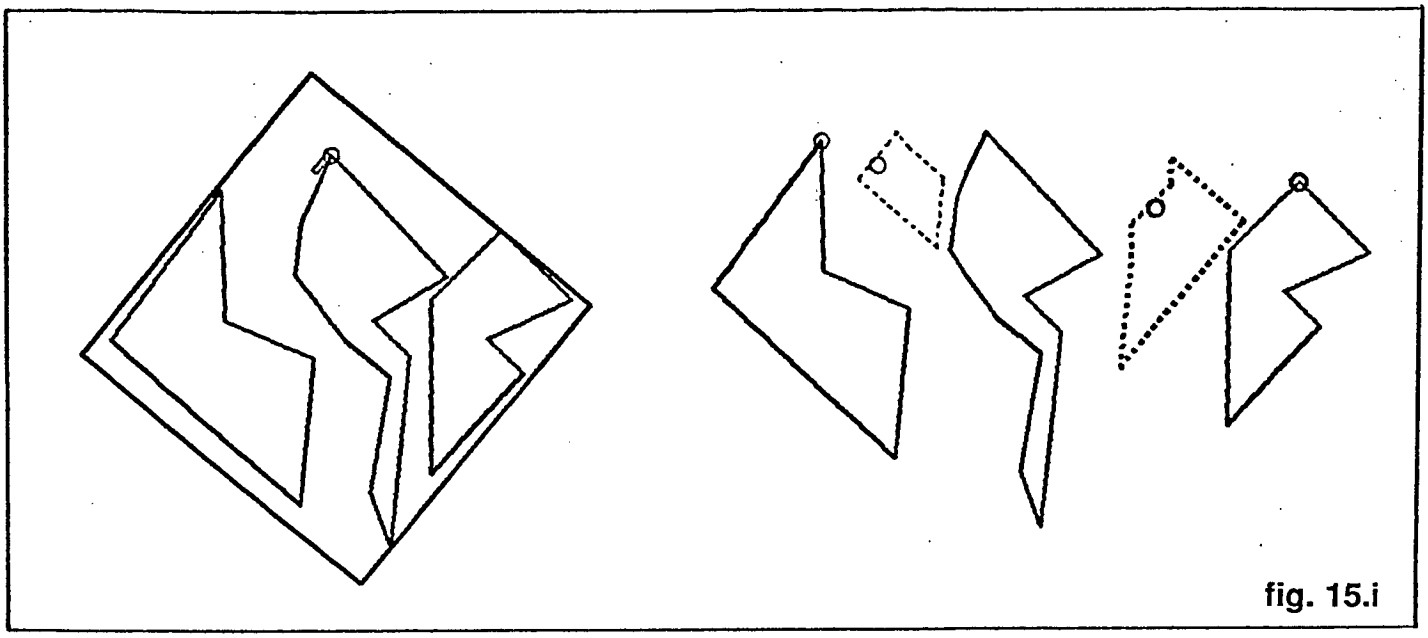

fig. 15.i

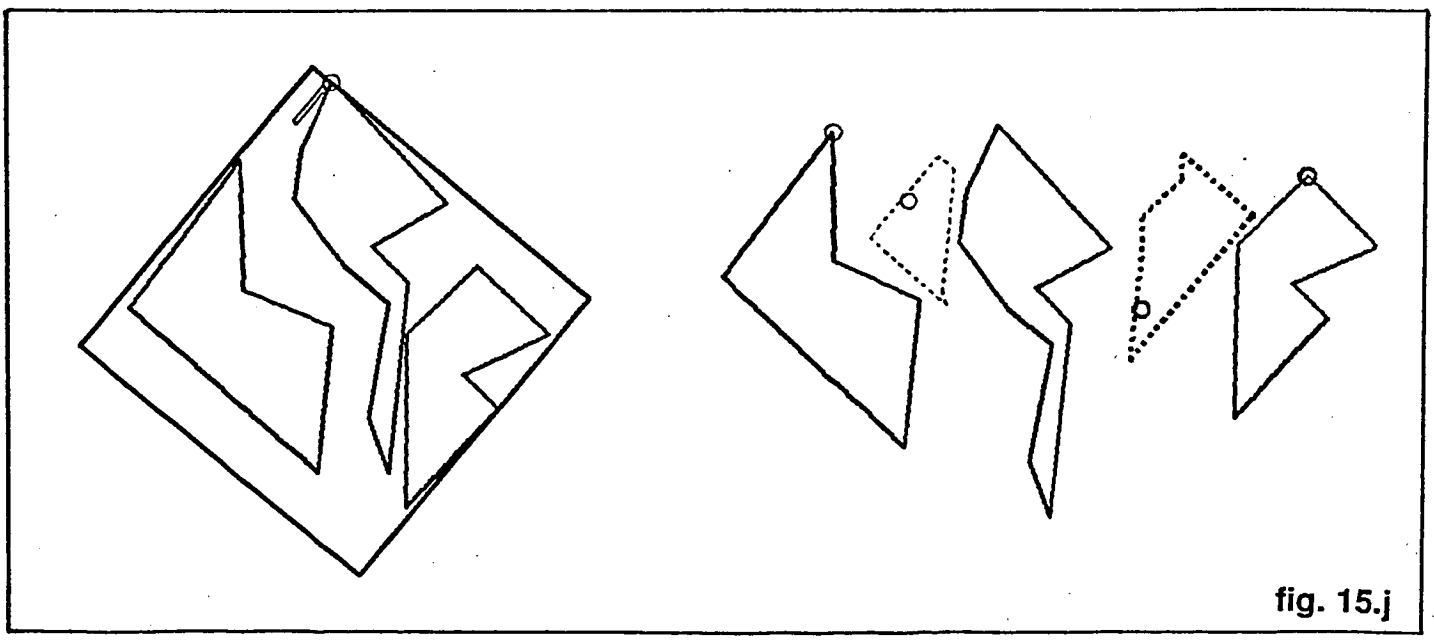

fig. 15.j

## 9- REFERENCES


(AA) Adamowicz, M., Albano, A., A solution of the rectangular cutting stock problem, IEEE Trans. Systems Man and Cybernetics, Vol. SMC-6, No. 4, 1976.

(AS) Albano, A., Sapuppo, G., Optimal Allocation of Two-Dimensional Irregular Shapes Using Heuristic Search Methods, IEEE Trans. Systems, Man and Cybernetics, Vol. SMC-10, No.5, May 1980.

(B) Boissonnat, J.D., Shape reconstruction from planar cross-sections, INRIA report No 546, 1986.

(BFM) Baker, B. S., Fortune, S. J., Mahaney, S. R., Polygon Containment under Translation, Journal of Algorithms 7, 532-548, 1986.

(C) Chazelle, B., The polygon containment problem, Advances in Computing Research, Vol.1, pp. 1-33, JAI Press, 1983.

(DBB) Dori, D., Ben-Bassat, M., Efficient nesting of congruent convex figures, Communications of the ACM., March 1984, Vol. 27, No. 3.

(F) Fortune, S.J., A fast algorithm for polygon containment by translation ICALP 1985.

(FZ) Favard,D., Zissimopoulos, V., Le problème de découpe, les difficultés, les méthodes, Rapport de Recherche n° 186, Université de Paris Sud, LRI, Décembre 1984. (in French)

(GG1) Gilmore, P.C., Gomory, R.E., Multistage cutting stock problems of two and more dimensions, Operations Research Vol 13 Jan-Fev.,1965.

(GG2) Gilmore, P.C., Gomory, R.E., The theory and computation for knapsack functions, Operations Research Vol 14 1966.

(GG3) Gilmore, P.C., Gomory, R.E., A linear programming approach to the cutting stock problem, Operations Research Vol 9 1961.

(GRS) Guibas, L., Ramshaw, L., Stolfi, J., A kinetic framework for computational geometry, 24th IEEE Symp. on Foundations of Computer Science, 1983.

(GS) Guibas, L., Seidel, R., Computing convolutions by reciprocal search, 2nd ACM Symp. on Computational Geometry, Yorktown Heights, 1986.

(H) Herz, J.C., A recursive computational procedure for two-dimensional stock cutting, IBM Res. Dev. Vol 16, 1972.

(HF) Haims, M.J., Freeman, H., A multistage solution of the template layout problem, IEEE Trans. on Systems, Sci. and Cybernetics VOL SSC6, 1970.

(HM) Hertel, S., Mehlhorn, K., Fast triangulation of the plane with respect to simple polygons, Information and Control 64, pp. 52-76, 1985.

(HMM) Hertel, S., Mehlhorn, K., Mantyla, M., Nievergelt, J., Space-sweep solves intersection of two convex polyhedra elegantly, Acta Informatica 21, 501-519, 1984

(KS) Kedem, K., Sharir, M., An efficient algorithm for planning translational collision-free motion of a convex polygonal object in 2-dimensional space amidst polygonal onstacles, 1st ACM Symp. on Computational Geometry, pp. 75-80, 1985.

(LPW) Lozano-Pérez, T., Wesley, M.A., An algorithm for planning collision-free paths among polyhedral obstacles, Commun. Ass. Comput. Mach., vol. ACM 22 (Oct 1979), 560-570

(MMS) McMullen, P., Shepard, G.C., Convex polytopes and the upper bound conjecture, Cambridge University Press, 1971.

(NR) Nievergelt, J., Reingold, E., Binary search trees of bounded balanced, SIAM J. Comput., Vol 2, No. 1, March 73.

(MS) Mehlhorn, K., Simon, K., Intersecting two polyhedra one of which is convex, research report, Fachbereich 10, Informatik, Universität des Saarlandes.

(OWW) Ottman, T., Widmayer,P, Wood, D.,, A fast algorithm for boolean mask operations, Computer Vision, Graphics and Image Processing, Vol. 30, 1985, pp. 249-268.

(PS) Preparata, F.P., Shamos, M.I., Computational Geometry, an introduction,Springer Verlag, 1985.

(S) Sharir M., Efficient Algorithms for Planning Purely Translational Collision-free Motion in Two and Three Dimensions, IEEE International Conference on Robotics and Automation, Rayleigh, 1987, pp 1326-1331.

(SS) Schwartz, J.T., Sharir, M., On the piano movers problem: III. Coordinating the motion of several independent bodies: the special case of circular bodies amids polygonal barriers. Courant Institute Technical Report 52, 1982.

(W) Whitesides, Sue H., Computational geometry and motion planning. Machine Intelligence and Pattern Recognition Vol 2, Computational Geometry North Holland 1985 pp 377-428

(WW) Widmayer, P., Wood, D., A time and space optimal algorithm for boolean mask operations for orthogonal polygons, Universitat Karlsruhe, Bericht 161, January 1986.

(Y) Yap, C.K, Coordinating the motion of several disks, Courant Institute Technical Report 105, 1984.

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique